FACULTAD DE INFORMÁTICA

UNIVERSIDAD POLITÉCNICA DE MADRID

TESIS DE MASTER

MÁSTER EN TECNOLOGÍAS DE LA INFORMACIÓN

DISTRIBUTED AWARD PROTOCOL: A COOPERATION AND COMMUNICATION METHOD FOR ROBOTS

AUTOR: KRZYSZTOF KWIATKOWSKI

TUTOR: NIK SWOBODA

JULIO 2013

Abstract

In this Master's Thesis a new Distributed Award Protocol (DAP) for robot communication and cooperation is presented. Task assignment (contract awarding) is done dynamically with contracts assigned to robots based upon the best bid received.

Instead of having a manager and a contractor it is proposed a fully distributed bidding/awarding mechanism without a distinguished master. The best bidding robots are awarded with contract for execution. The contractors make decisions locally. This brings the following benefits: no communication bottleneck, low computational power requirement, increased robustness.

DAP can handle multitasking. Tasks can be injected into system during the execution of already allocated tasks. As tasks have priorities, in the next cycle after taking into account actual bid parameters of all robots, tasks can be re-allocated. The aim is to minimize a global cost function which is a compromise between cost of task execution and cost of resources usage.

Information about tasks and bid values is spread among robots with the use of a Round Robin Route, which is a novel solution proposed in this work. This method allows also identifying failed robots. Such failed robot is eliminated from the list of awarded robots and its replacement is found so the task is still executed by a team. If the failure of a robot was temporary (*e.g.* communication noise) and the robot can recover, it can again participate in the next bidding/awarding process.

Using a bidding/awarding mechanism allows robots to dynamically relocate among tasks. This is also contributes to system robustness.

DAP was evaluated through multiple experiments done in the multi-robot simulation system. Various scenarios were tested to check the idea of the main algorithm.

Different failures of robots (communication failures, partial hardware malfunctions) were simulated and observations were made regarding how DAP recovers from them. Also the DAP flexibility to environment changes was watched.

The experiments in the simulated environment confirmed the above features of DAP.

Resumen (español)

En la Tesis Fin de Máster, se presenta un nuevo Distributed Awarded Protocol (DAP) para la comunicación y la cooperación entre los robots. La asignación de tareas (adjudicación del contrato) se realiza de forma dinámica con los contratos que mejor se adapten a los robots de licitación/subasta.

En lugar de tener un gerente y un contratista se propuesto que haya un mecanismo de adjudicación sin un líder. A los mejores robots de la subasta se les otorga la ejecución el contacto. Los contratistas toman las decisiones en nivel local. Esto trae las siguientes ventajas: no hay atascos de comunicación y el requerimiento de cálculo es bajo y se incrementa robustez.

DAP puede manejar múltiples tareas. Las tareas pueden ser inyectadas en el sistema durante la ejecución de otras tareas ya asignados. En el siguiente ciclo, tras tener en cuenta los parámetros de la oferta de los robots, las tareas pueden ser reasignadas. El objetivo es minimizar la función de costes globales, un compromiso entre el coste de la ejecución de las tareas y el coste de los recursos usados.

La información sobre los valores de la tarea y la oferta se reparte entre los robots con el uso de una ruta de Round Robin que es una solución novedosa propuesta en este trabajo. Este método permite también la identificación de los robots fallidos. Tales robots no se eliminan de la lista de robots adjudicados y su sustitución se encuentra, por lo que la tarea aún se ejecutada por un equipo. Si el fallo del robot era temporal (por ejemplo, ruido de comunicación) y el robot puede recuperarlo, después puede participar en la próxima nueva oferta / proceso de adjudicación.

Usar una oferta / mecanismo de adjudicación permite a los robots reubicarse dinámicamente entre las tareas. Esta es también la contribución a la robustez del sistema.

DAP se evaluó a través de múltiples experimentos realizados en el sistema de simulación de múltiples robots. Varios escenarios se analizaron para comprobar la idea del algoritmo principal.

Se simularon diferentes fallos de robots (fallas en la comunicación, fallos de hardware parcial) y las observaciones se realizaron viendo cómo DAP se recupera de ellos. También se observó la flexibilidad DAP a los cambios del entorno.

Los experimentos en el entorno simulado confirmaron esas características de DAP.

Table of content

Table of figures

Table of tables

1 Introduction

The Distributed Award Protocol (DAP) is a proposed model for robot communication and cooperation. It is based on the Contract Net Protocol [45] with implemented modifications. Choosing CNP as a base platform brings the opportunity of task assignment (contract awarding) dynamically with contracts best suited to bidding robots. Using bidding/awarding mechanism allows robots dynamically relocate among tasks. This is also the contribution to system robustness.

The modification is that instead of having a manager and a contractor (in Murdoch [19][22][23]: auctioneer and bidder) there is a fully distributed bidding process without a distinguished master. The best bidding robots are awarded by contact execution. The contractors make negotiation decisions locally.

This implies the following benefits: no communication bottleneck and low computational power requirements. The lack of a manager eliminates a sensitive point for robustness: if the manager fails then all team of robots is inoperable.

DAP assumes that in the system there is no a delegated robot playing the role of an auctioneer. All involved robots are self-disciplined and honest. They communicate with each other about their actual potential to fulfill the task. Then they individually decide to which of them the task will be assigned.

As each robot knows the bid values of its colleagues and holds in the local memory the rules of awarding the task, it is not possible that a robot is not assigned to a task (even having all required qualifications to fulfill the task) due to arbitral auctioneer's decision.

DAP can handle multitasking. Tasks can be injected into system during the execution of already allocated tasks. Tasks have priorities so in the next cycle after taking into account actual bid parameters of robot can be re-allocated. The aim is to minimize a global cost function which is a compromise between cost of task execution and cost of resources usage.

The global cost is calculated as the sum of the individual robot cost, and the goal is to complete the all tasks within minimized time of robots working. Additionally, when robots are not allocated to any task they remain stationary at their current positions (being in the listening mode) for energy saving.

DAP is a resource-centric protocol, for example it optimizes the energy consumption during message broadcasting. In Murdoch [23] a broadcast is always done using the full-power transmission range. Here the transmission power is increased step by step until the nearest neighbor is found and then the signal strength is memorized and used for later return transmission.

Information about tasks and bid values is spread among robots with the use of Round Robin route. This method allows the identification of failed robots. Such failed robots are eliminated from the list of awarded robots and replacements are found so the task is still executed by a team.

If the failure of robot was temporary (*e.g.* communication noise) and the robot can recover it can again participate in the bidding/awarding process.

In the next chapters all these features will be discussed in details preceded by explanation of DAP algorithm.

2 Related work

There are different models for robots cooperation starting from fully centralized to fully distributed models.

In centralized solutions ([8][14][24][31][40]) there is a delegated robot playing the role of manager. All team robots report to it and the manager gives commands to the team members regarding how to proceed. The benefit is the optimal planning of activities, on the others side it is the critical point of failure limiting robustness. Moreover, if the environment changes rapidly the team members report these changes to the manager thus there are two negative issues. The first is that the manager may not have sufficient computational power to find solutions for such rapid changes. Large amounts of information put high bandwidth requirements on the communication, too because all team members are to be in contact with manager all the time. If manager fails the all team of robots is inoperable. Due to these reasons the centralized model is recommended for small teams of robots operating in static environment.

On the other hand there is the a fully distributed approach ([1][2][3][7][15][17][33][45][48]). Many of these models are bio-inspired. The main difference is that each robot locally plans its own activity so little computation is needed. There is no global information available to robots so the decisions are made only based on local information. This is the reason that task planning and execution is not optimal. The advantage is that each robot has implemented a simple algorithm, unlike the complex algorithms of the centralized approach. As robots communicate with others in their neighborhood only there is no strong requirement for communication capacities. As compared to centralized models, distributed models are typically faster, more robust, and more flexible to changes. However, they are far from the optimal solution.

In the middle there is a compromise for above approaches. It is a market-based model ([10][13][23][31][32][39][49][50][51]). The general concept is that the robots act largely independently in terms of planning for themselves, but are able to take into account team resources. This is done by trading tasks with team members. Task execution is preceded by negotiations and task allocations. During this phase each robot announces all of its current tasks for auction. The award is given to the task corresponding to the highest bid it receives.

Communication is limited to bids and awards of tasks, so there is a requirement for low-bandwidth communication only. However, these economic-like models are not without their disadvantages, because some negotiation schemes can drastically increase communication requirements.

DAP combines all advantages from those three models: centralized, distributed and market-based. In DAP all robots have the same global information about tasks and other robots' bid values as leader in a centralized approach. This allows allocating tasks to robots in the optimal way.

From distributed model is taken that robot communicate only with their neighbors so broadcast requirements are low. Thus problems with communication bottleneck to contact the leader are avoided. In distribution model a little computation is required and each robot has enough computation power to allocate task by itself as per algorithm written in the local memory.

From the market-based approach the bidding mechanism is adapted what enables robot to make decisions with include of team resources. This is also a contribution factor to increase robustness.

The novel approach presented here is that after task injection by an initiator it does not play the role of manager. Robots communicate with each other by passing information in a Round Robin manner and award the best of them locally in the bidding process. In each repeated cycle the potential of the robots to execute a task and progress of the task execution is monitored. This information is all the time available to all robots. In each cycle the best robots are awarded the task. Progress of the task execution is monitored – if some robots announce worse bid parameters they are excluded from task execution (still having the opportunity to be allocated if they improve) and their best replacements are allocated instead. When a task is accomplished it is reported to the initiator. So the initiator is not involved in details of task execution. It only delegates the task and receives the final report that task is executed.

During the bidding process the information regarding tasks and environment is dynamically passed to all robots in a Round Robin manner without going through a centralized hub. This not only eliminates the communication bottleneck but also contributes to flexibility over dynamic task injection/changes and dynamic environment.

Below there is a short description of different net protocols.

Contract Net Protocol

The Contract Net Protocol (CNP) [45] has been developed to specify problem-solving communication and control for nodes in a distributed problem solver. CNP is a high level protocol supporting communication among agents in distributed multi agent system.

There is a collection of nodes introduced referred to as a contract net. Task execution is considered as a contract between two nodes. Each node can play a role of manager or a contractor. A manager monitors the execution of a task and processes the results of its execution. A contractor is responsible for the actual execution of the task. This role can change dynamically during the course of problem solving.

Task allocation is done through a negotiation process which is controlled by manager. At any time, any robot can be a manager or a bid contractor thus allowing subcontracting. A discussion is carried on between nodes with tasks to be executed and nodes that may be able to execute those tasks.

Distributed model of control increases reliability and allows slightly degradation of performance in the case of individual node failures. Due to the distributed architecture there are no nodes whose failure can completely block the contract negotiation process. Additionally, recovery from the failure of a node is also achieved because the manager can detect the failure of any of its contractors and then again announce the bid and award another contractor.

CNP protocol can be used in applications requiring such control and data distribution to avoid bottlenecks and a finer degree of control in making resource allocation.

The features of CNP can be summarized as below:

- Tasks are assigned (contracts awarded) dynamically, resulting in the better deals for the parties (agents) involved.
- Agents can enter and leave the system at will.
- The tasks will be naturally balanced among all the agents since agents that already have contract(s) don't have to bid on new ones. If an agent is already using all its resources, it will be unable to bid on new contracts until the current ones are completed.
- A reliable strategy for distributed applications with agents that can recover from failures (to be discussed more in the following paragraphs).

DAP also uses a negotiation mechanism called bidding process.

However this process does not have a manager controlling the negotiations. Bidding/awarding is done locally.

Robots locally compare bid parameters and locally decide which of them will be awarded as all of them have the same global information.

FIPA CNIP

Foundation for Intelligent Physical Agents (FIPA) [18] recommended the Contract Net Interaction Protocol (CNIP) as a modified version of the original CNP for agents' communications [16].

CNIP standardized relations between Initiator Agents (IA) and Contractor Agents(CA).
To get contract awarded there are four steps to go:

- The initiator sends out a Call for Proposals (CFPs) to all possible contractors.
- Each contractor reviews the received CFPs and bids on the most feasible contracts. This is to be done within a specified period of time.
- The initiator chooses the best bid and awards the contract to the bid winner
- Action is closed and initiator rejects other bids.

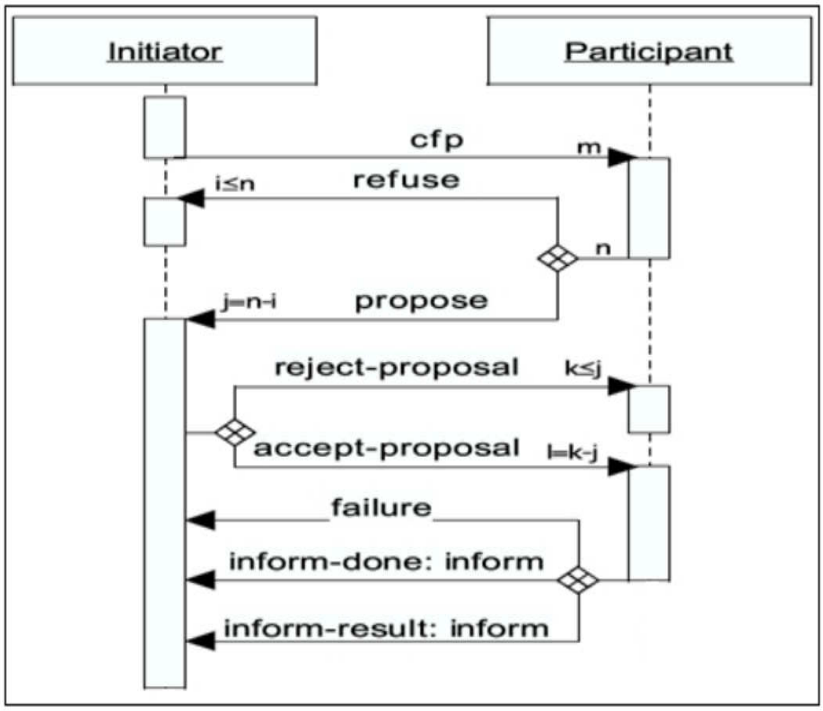


Figure 1: FIPA Contract Net Interaction Protocol

The similarities and differences between DAP and FIPA CNP are following:

- in DAP there is no distinguished the initiator, it is responsible only for task injection – it does not monitor the task execution – only waits for the final report that task is done

- in DAP each robot talks only to its neighbor (via Round Robin message passing all robots share the same information), in CNP all robots talk to the initiator
- in DAP there are no refusals, all robots have to participate in the bidding, the non-explicit refusal is in the case when robot failures
- in DAP the "ACK: sent forward" message is introduced signaling that robot sent the information to the next robot. This prevents from the break of Rood Robin passing information through robots thus increasing robustness.

ALLIANCE

ALLIANCE [38] is fault-tolerant and adaptive multirobot coordination architecture for physically embodied robots. This is a behavior-based system with the added benefit of motivational behaviors such as impatience and acquiescence. These features enable robots to complete tasks when other robots fail to execute them and to give up on tasks they cannot complete. Every robot knows the entire task to be completed. Each robot tracks both its own and other robots' potential to execute a task and a task execution progress. Based on this information robots calculate the values of impatience and acquiescence. If a robot becomes sufficiently impatient, it may overtake a task from a sufficiently acquiescent robot. This is the solution for providing fault tolerance of the system. These motivational behaviors allow robot teams to be fault tolerant and adaptive. However this approach does not enable robots to respond to dynamic conditions in a quick and optimal manner. Also robots do not reason about the limited resources available to them and attempt to optimize utilization. Moreover no allocation of tasks is performed in this architecture. Instead, high-level tasks are programmed into the behavior sets of the robots. This scheme does not promote optimized task allocation, and does not allow new types of tasks to be dynamically assigned.

DAP is also a fault-tolerant system. This is achieved by frequent bidding using Round Robin method. This feature in addition allows for dynamic task injection and optimized tasks allocation in contrary to the ALLIANCE.

M+ Protocol

The M+ Protocol [3] is based on an adapted version of the Contract Net Protocol for the negotiation. It is a decentralized multi-robot cooperative mechanism which uses negotiation techniques. This allows a robot to choose incrementally the best task to be executed taking into account the current context. System is based on local planning, task negotiation and cooperative reaction to failures during task execution. The M+ cooperative reaction activity is invoked when a failure occurs during task execution. Then M+ updates the world state, manages the exchange of information between the robots and re-plans the activity. Tasks are allocated and re-allocated when necessary through the negotiation process. When, for some reason, a task execution fails, the robot first tries to find another set of actions to attain the

same goal. If it fails, it requests help from other robots to find plans of cooperative actions that satisfy both robots plans. M+ is a cooperative scheme for task allocation and has an interface to higher level task planner. According to its context and capabilities, robots plan their missions by considering other robots' plans as constraints. These plans contain coordinated and co-operative actions, which together with task cost estimation and anticipate knowledge of the next task, serve as basis for negotiation.

DAP also can identify a failed robot, de-allocate it from the task and find the best suited robot to allocate it to task as a replacement and thus continue task execution. The difference between M+ and DAP is that here is no leader and information is passed in Round Robin. No presence of leader gives advantage of increased robustness and no communication bottleneck.

Murdoch

MURDOCH [22][23] is a distributed, resource-centric communication model. The communication method between agents introduced in this approach is based on subject addressing known as Publish/Subscribe. Robots interact by broadcasting anonymous messages. These messages have one sender and several recipients. Instead of sending a message to a given destination (destination-based addressing), messages are broadcasted with one or more tags that relate to their content (subject-based addressing). The members of the team which are interested in these contents receive automatically these messages by subscribing them. Thus, this method of communication is named Publish/Subscribe. Using it, heterogeneous and individually autonomous robots can communicate and cooperate efficiently, through a decentralized manner, in a dynamic and unpredictable environment, better exploiting the capabilities of these robots.

A key feature in this approach is that all communication is resource-centric, and never name-based. Thus, the claim is that all messages are addressed in terms of the resources required to do the task. All tasks are allocated based on a single-round auction scheme. The auctioneer determines a winner and notifies the bidders. The winner is awarded a time-limited contract to complete the task. The auctioneer is responsible for monitoring the progress of the task execution. To do this, the auctioneer periodically sends contract renewal messages to the winner, which sends back corresponding acknowledgements. These messages will have to be addressed by name and will increase the communication requirements of the system since the auctioneer and winner will have to remain within communication range (or periodically return to positions within communication range) to renew the contract. Furthermore, since the auctioneer assumes a fault if a renewal message is not acknowledged and reassigns the task, several robots could attempt to complete the same task if acknowledgements are not received on time or some acknowledgement messages are lost.

Robot playing a role an auctioneer may centralize the system. However, any individual may act as an auctioneer, therefore the system becomes distributed. In terms of fault-tolerance, the limited-time contract to execute tasks is very important. The auctioneer may assume that a robot has failed or is executing too slowly and may assign the task to another robot.

Murdoch is a distributed system, where the negotiations are controlled by auctioneer. DAP is also a distributed system, where the negotiations are done locally without involvement of manager/ auctioneer.

Murdoch is limited to one task only, whereas DAP allows for dynamic task injection/allocation.

In Murdoch a broadcast is always done using the full-power transmission range. Here the transmission power is increased step by step until the nearest neighbor is found and then the power level is memorized and used for later return transmission.

DEMiR-CF

DEMiR-CF [43] is a framework using bid evaluation for dynamic task selection and enabling task re-allocation when profitable. Auction model is used by robots to declare intentions on task execution and then select of the best bidders for task execution. The modification is an introduction of coalition leader. Each robot can offer a new auction and create coalition with other robots to execute a task. When a robot finds out its cost to be lower than the maximum cost of the coalition it sends a join request to the leader of coalition. The coalition leader can add and release robots from coalition (due to low bid value or failure).

If there is a sufficient number of members in a coalition to execute a task the surplus robots can be released from coalition and announce their intentions for executing another task. It is coalition leader responsibility to broadcast the maximum cost of execution for all coalition members in each execution step. And it has an influence on the robustness of the system. If coalition leader fails then the all robots from the coalition are inoperable.

In DAP a negotiation process is also used but there is no leader. Bidding/awarding is distributed and done locally by all robots which pass to each other the information about their bid values thus eliminating the possibility of the team fail due to leader fail.

In addition a Bid_Table is transparent to all robots which gives a global overview for each robot same as for leader in DEMiR. In DEMiR there are coalitions of robots with leader, only who possess the global information. In DAP there is a team of robots without a leader – and each of robot is in possession of global information.

3 Constraints and assumptions

The definition of the environment is following:

- Number of robots
  set of n robots $\Phi = \{R_1,\ldots,R_n\}$
- Types of robots – homogeneous
- Each robot knows its own geographical coordinates in the area
- Number of tasks
  set of m tasks $\Lambda = \{T_1,\ldots,T_m\}$
- Tasks have different priorities
- Area of operation - constrained
- Type of area – dynamic changing parameters
- Initiator has the same communication capabilities as robots,
  Is located outside the operational area
- Communication – bidirectional
- Robots are identified by unique number *Robot_ID*
- The signal power for communications can cover all the area
- The signal power increases in steps

Each activity is to be done within a specified period of time.

The following time intervals are defined globally and known by all robots

- *waiting_for_task_request_time*
- *waiting_for_aknowlegment_time*
- *contract_exeuction_time*

Relations between these times are shown on Figure 2. Time intervals are represented by rectangles.

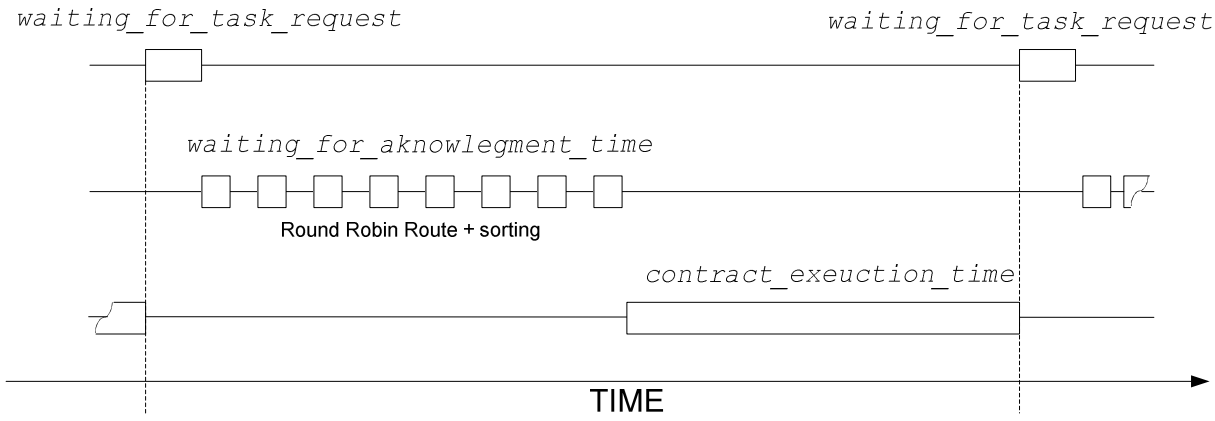


Figure 2: DAP timing

4 Distributed Award Protocol description

DAP is a protocol that performs the specific task of regulating the data transmission between

a set of *n* robots $\Phi=\{R_1,\ldots,R_n\}$ having to execute

a set of *m* tasks $\Lambda=\{T_1,\ldots,T_m\}$.

Having a set of *n* resources and a set of *m tasks*, the goal is to assign resources to tasks in an efficient manner with the focus on minimizing resources usage and task completion time.

At the heart of DAP lies a protocol that allocates tasks via a one-shot highest bidder auction. The approach of this algorithm is to minimize a global cost function (resource usage and time to execute a task).

Tasks have priorities and can be inserted into system dynamically – that means during the execution of already assigned tasks. The message that a new task is added is communicated to all robots in a Round Robin way. Then robots locally sort all tasks by priority and calculate bidding values for each task. Next, based on bid values, robots individually make decisions which of them are awarded to execute a task.

Using a Round Robin algorithm has two basic benefits. The first is that all the robots will be queried for their potential to fulfill the task. The second is that the identification and elimination of failed robots (reasons of the failure can be different – for instance communication error, mechanics or hardware malfunction errors).

The description of DAP is divided into three chapters: single task, multi tasks and exceptions handling. The reason of such approach is to present the main idea of protocol in the easiest way and then to describe extension to multitasking. At the end the way of handling exceptions will be presented - such as communication failures and task allocation bottlenecks. This structure should fluently introduce a reader to the Distributed Award Protocol.

4.1 Single task

Task is injected by an initiator. It can be a human operator or computer system or a robot. Physically it is located outside of the area where robots operate. The initiator has the same communication and measurement capability as all involved robots – it can be treated as a robot R_{-1}.

It sends the initiating message - *"I have a task".* The initiator keeps increasing range of its signal to get an acknowledgement from the nearest robot R_c where

$c \in [1..n]$. An acknowledgment message carries *Robot_ID* of robot R_c. If there is more than one robot in the same circular area, the first which sent the acknowledgement is chosen.

Having received acknowledgment the initiator responds with sending to R_c information about the task requirements - *Task_Specification* containing task specific information. An example for a search/rescue task would be target coordinates and number of robots required for the task.

After receiving this data the robot R_c does not send a return acknowledgment at that moment. It will send the acknowledgment after forwarding this data to the next robot R_{c+1}. This is aimed to reduce communication steps and increase a speed of data transmission.

Each of the n robots has a *Bid_Table* with columns related to fields of the *Task_Specification* table. The table at start up is filled with zeros. Robot R_c calculates a *Bid_Value* based on *Task_Specification* and its parameters (for example: the actual position of the robot and a resource usage to execute the task - detailed calculation will be described later on). It puts *Bid_Value* in the row in the *Bid_Table* indexed by its *Robot_ID*.

Then robot R_c has to identify the nearest robot R_{c+1} to pass to it the information about the task. So R_c keeps increasing range of its signal to get an acknowledgement from the nearest robot R_{c+1}. An acknowledgment message carries the name (*Robot_ID*) of robot R_{c+1}. Robots, which already have bid tables for the given task, do not make any acknowledgement. Only robots, which are waiting for tables are in the listening mode.

Then robot R_c sends a message to robot R_{c+1}. It contains the information about task – *Task_Specification*. This information is needed to calculate the *Bid_Value* of the addressed robot R_{c+1}. Robot R_c also sends *Bid_Table* with bidding information from all previous robots (to speed up a transmission only non-zero rows are sent).

Robot R_{c+1} repeats the steps in localizing the next robot R_{c+2} and sends to it the *Task_Specification* and *Bid_Table* (data from previous robots updated by data from R_{c+1}) and only then it responds with acknowledgment to robot R_c.

If this acknowledgment from robot R_{c+1} does not arrive to robot R_c within the *waiting_for_aknowlegment_time* that means a failure has arisen. Detailed description of errors handling is in the chapter "Exceptions to Protocol Flow".

These steps are repeated in a Round Robin manner until all robots receive the *Bid_Table*. However, only the last robot R_L has information about all bid values from its n - 1 colleagues. So the fully filled *Bid_Table* is sent back to all robots also using the Round Robin manner. Here the data contained in columns *From* and *Signal_Strength* are useful. Point-to-point communication is used, where the robot sends information to a neighbor indexed by value *From.* It stops when the robot which started bidding receives the *Bid_Table* filled with data from all n robots.

The advantage of this method is that during the second pass of information in a Round Robin route robots know how big Signal_Strength is to be applied to send message to the next robot. That gives a resource optimization advantage.

Now all n robots have the same data – a fully filled *Bid_Table* with data from all n robots. They sort the table individually by column *Bid_Value*. A decision is made locally regarding if their *Robot_ID* is included in the w top table positions of the *Bid_Table*, where w *is* the number of the required robots for the task. Figure 3

In such situation robot R_A (awarded) – where $a \in [1 \ldots w]$ proceeds with task execution; otherwise it stops and enters into the listening mode. The number of awarded robots should be equal to w. If a is greater than w (there are more aspiring robots) there is an exception. The choice algorithm will be discussed in chapter dealing with exceptions.

Now there is a phase of contract execution. The awarded robots proceed with task execution. This phase lasts over *contract_exeuction_time.* When this time passes the next bidding process is repeated again. The role of initiator is now played by the robot R_V with the highest score in previous bidding**.** Then new bidding starts with $R_c=R_V$. If robot R_A fulfills the task then checks its column *Done* in the *Bid_Table*. If all w robots check the column *Done* then the task is fulfilled – mission accomplished.

Task accomplished report

When the last robot from awarded robots executed a task then it sends a report to the initiator. This is done by sending a full power signal with the message "*Task_done*".

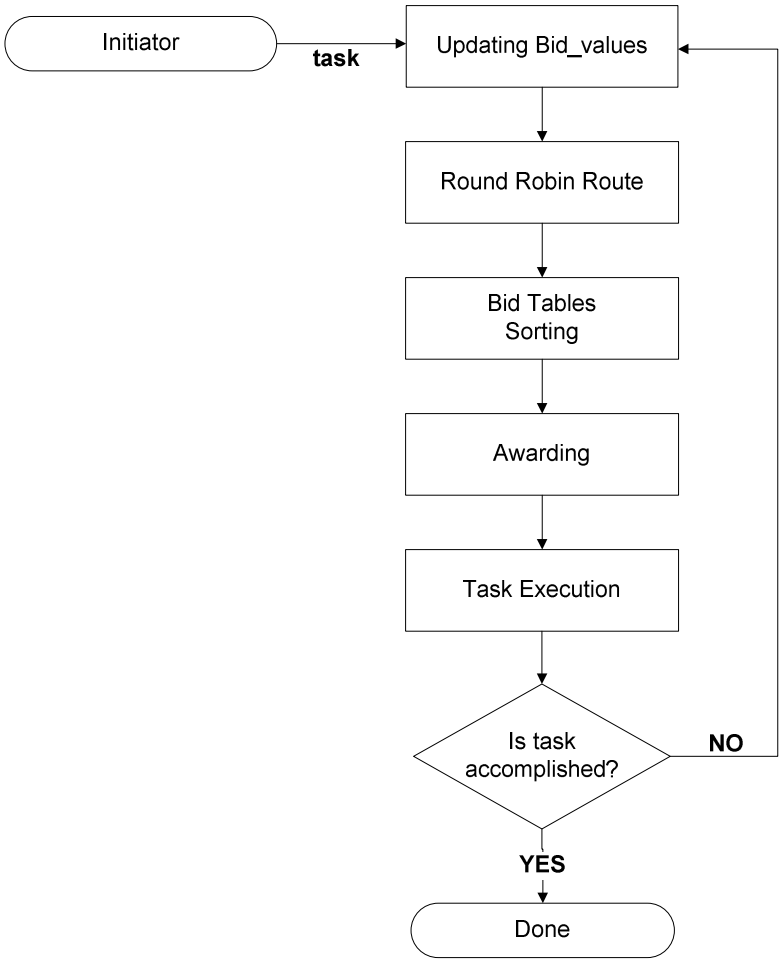


Figure 3 Single task in DAP

Figure 4 presents the information flow along the time axis. The rectangles represent the states of three cooperating robots R_{c-1} R_c R_{c+1}. At the beginning robot R_c is in the listening mode. After receiving *Task_Specification* and *Bid_Table* it forwards this data (updated by its inputs) to the next robot R_{c+1} and then responds with acknowledgment to robot R_{c-1}.

In the Return Round Robin Route each robot knows the *ID* of its neighbor so this route is much faster because there is no need for the phase of neighbor seeking by increasing signal strength in steps what is done during the First Round Robin Route.

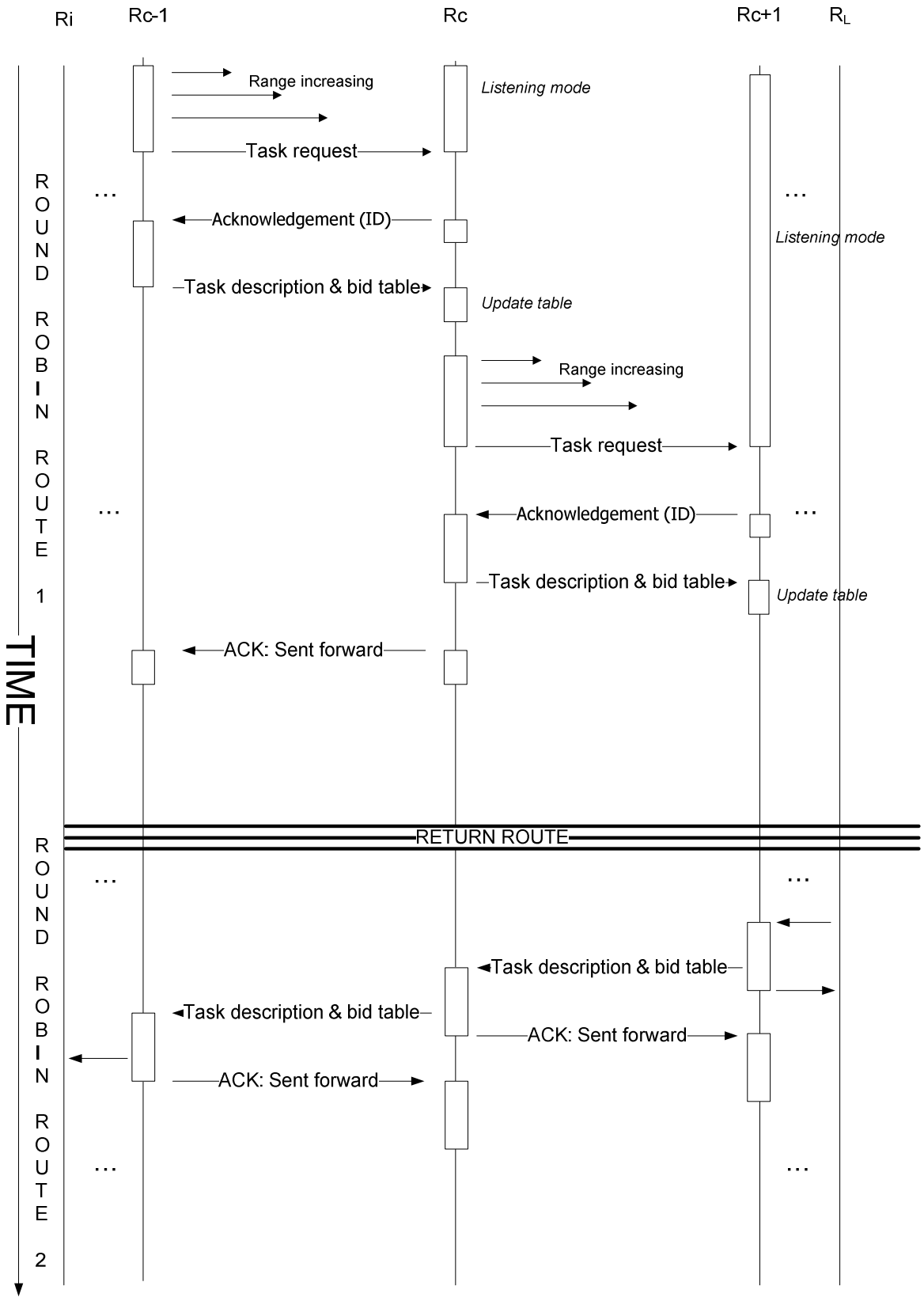


Figure 4: DAP in the time domain

4.2 Multi-task allocation

The previous chapter described the bidding/awarding activity for one task only. Now the situation with multiple tasks will be considered. As it has already been mentioned robots communicate with each other by sending *Task_Specification* and *Bid_Table*.

In this extension the initiating message - "*I have a task"* for a single task changes into "*I have a task numbered Task_ID*". This modification enables accepting and processing multiple tasks. In the listening mode robots are not "deaf" to incoming task requests even if they are already involved in execution of earlier assigned tasks.

Multiple tasks allocation requires introducing additional *Task_Table.*

The *Task_Table* contains the information about the set of *m* tasks $\Lambda = \{T_1,\ldots,T_m\}$. Each element of this set consists of *Task_Request* (which is an extension of *Task_Specification* for a single task) indexed by *Task_ID*.

Task_Request consists of three elements: *Task_ID, Task_Priority* and *Task_Specification*.

Robot R_c starting the cycle has to identify the nearest robot R_{c+1} to pass to it the information about the tasks. So R_c keeps increasing range of its signal to get an acknowledgement from the nearest robot $R_{c+1.}$ An acknowledgment message carries the name (*Robot_ID*) of robot R_{c+1}.

Then robot RC sends two packets of information to robot R_{c+1}. The first block is the information about task - *Task_Request* (*Task_ID, Task_Priority* and *Task_Specification*). This information is needed to calculate the *Bid_Value* of the addressed robot R_{c+1}. The second block is the *Bid_Table* with bidding information from all previous robots (to speed up a transmission only non-zero rows are sent).

Robot R_{c+1} repeats the steps in localizing the next robot R_{c+2} and sends to it the Task_Request and *Bid_Table* (data from previous robots updated by data from R_{c+1}) and only then it responds with acknowledgment to robot R_c.

In the case of multiple tasks each task has a separate *Bid_Table*. The difference now is that the *Task_Table* is sorted first by *Task_Priority* and then the *Bid_Table* related to *Task_ID* is sorted in the way described for a single task.

The highest *Task_Priority* column is sorted and the awarded robots for this task are excluded from bidding process for other tasks of lower priority.

The awarded robots are excluded only for the current bidding. In the next bidding process they will participate as usual. This gives a flexible reaction to dynamical environment and to dynamic allocation of injected tasks with aim to optimize costs and resource usage.

After the awarding phase there is a contract execution phase and only then a window is open to accept new task request (*waiting_for_task_request_time).* New task requests are not admitted during the bidding process.

To speed up information exchange not the complete *Bid_Table* is sent for each task. The solution is to extend *Bid_Value* from one numeric value into one-dimensional array, where each element holds the *Bid_Value* for each task. Robot puts *Bid_Value* to appropriate cell

in this array indexed by *Task_ID.* Such a modified *Bid_Table* is communicated to other robots by Round Robin.

The *Task_Table* is sorted by priority. Robots take the highest priority *Task_ID* and sort *Bid_Table* related to this task and allocate the required number of robots. Then the start index of sorting *Bid_Table* is increased to the sum of required robots from last allocated tasks. Robot takes another task with lower priority and refreshes *Bid_Value* column. Robot sorts *Bid_Table* from start index (which is now not 0, but required robots of the highest priority task). Then available robots are allocated to tasks of lower priority. The sorting process is finished when all tasks have allocated robots. Figure 5

When *Task_Priority's* or *Bid_Value's* are equal after sorting, then there is an exception. The solution is based on the main assumption that robot can be assigned to only one task. For both situations the robots with the highest *Task_Priority* or respectively *Robot_ID* are chosen.

```
Public Sub sort_bid_table_partial(ByVal task_index As Byte)
   Dim br As Bid_Row
   Dim tr As Task_Row = Task.arrTasks(task_index)._task_row
   bid_table_end_index = ile_robot + 1 - bid_table_start_index

   For r = bid_table_start_index To ile_robot
      If bid_table(r).done = False Then bid_table(r).robot.allocate_task = Task.arrTasks(0)
      bid_table(r).calculateBidValue(bid_table(r).robot, Task.arrTasks(task_index)._task_row)
   Next

   bid_table.Sort(bid_table_start_index, bid_table_end_index, comparer)
   bid_table.Reverse(bid_table_start_index, bid_table_end_index)

   end_ind = bid_table_start_index + tr.required_robots - 1
   If bid_table_start_index <= ile_robot And end_ind <= ile_robot Then
      For ind = bid_table_start_index To end_ind
         br = bid_table(ind)
         br.ComboBoxUpdateText(CStr(br.Bid_Value))
         br.index_sorted = ind
         br.robot.allocate_task = Task.arrTasks(task_index)
      Next
   End If
   bid_table_start_index += tr.required_robots
End Sub
```

Figure 5: *Bid_Table* sorting code

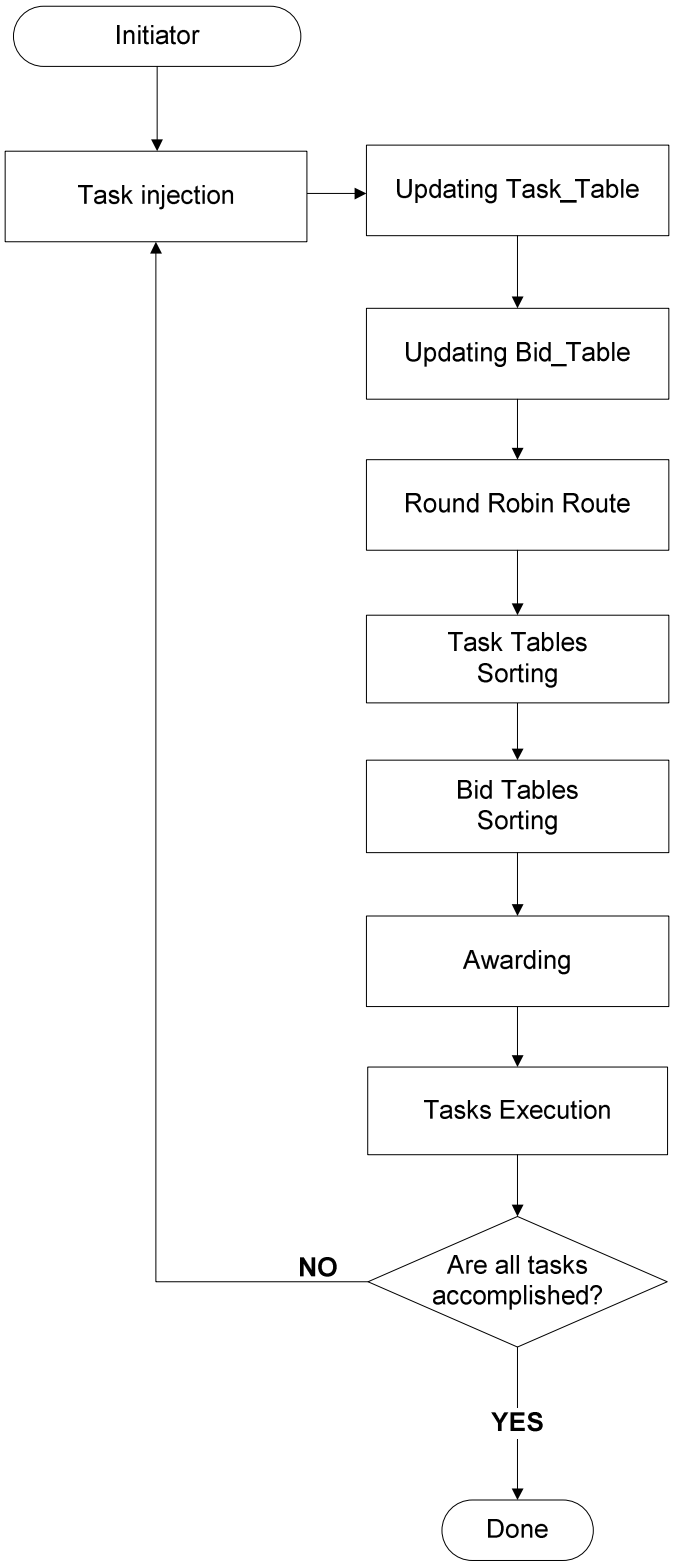


Figure 6: Multi-task allocation in DAP

The above Figure 6 is the illustration of expanding the single task DAP protocol into multitask allocation. The tasks are allocated and executed in parallel. Updating of *Bid_Table* is preceded by updating of *Task_Table.*

Task injection

Round Robin routes and tables sorting period are so much shorter than *contract_exeuction_time* that can be neglected in timing consideration of DAP. The cycle starts with initiator sending a first task message to the nearest robot R_c which initializes a Round Robin transmission. After receiving all data and tables sorting a phase of task execution starts. All robots have internal timer to measure it. Each robot launches this timer after sorting the *Bid_Table*. When this time is up a robot launches another timer counting down from *waiting_for_task_request_time.* The initiator also measured off *contract_exeuction_time* as it has the same communication and measurement capability as all involved robots – it can be treated as R_{-1} robot. If it has a new task to be added it sends the message "*I have a task numbered Task_ID*". The initiator keeps increasing range of its signal to get an acknowledgement from the nearest robot R_c. An acknowledgment message carries *Robot_ID* of robot R_c. If there is more than one robot in the same circular area, the first which sent the acknowledgement is chosen.

After *waiting_for_task_request_time* the next cycle of bidding will start. This cycle will be started by the robot R_v with the highest score in previous bidding**.** When the Round Robin turn will come to robot R_c which received the new task it will send the *Task_Table* extended by the new injected task. The knowledge about this task will be fully propagated to all robots in the next cycle of bidding.

Task accomplished report

When the last robot from awarded robots for particular task executed a task then it sends a report to the initiator. This is done by sending a full power signal. The initiator responses with a message carrying the information if the robot which executed task can be released for other tasks or have to stay idle. This corresponds to injection of the *Task_ID* with *Task_Priority*=0. Propagating task with zero priority through robots means for them to delete this task from the list in *Task_Table*. Information if the robots which accomplished task can be released or not is coded in the following way. If they are to be released their *Task_ID* in *Bid_Table* will be reset to 0, if not – the *Task_ID* of the just executed task will remain there.

4.3 DAP context

4.3.1 *Task_Table* description

The *Task_Table* structure is shown on the below figure.

Task_ID	Task_Priority	Task_Specification	Done
1	TP_1	TS_1	TD_1
2	TP_2	TS_2	TD_2
...	...	...	...
z-1	TP_{z-1}	TS_{z-1}	TD_{z-1}
z	TP_z	TS_z	TD_z
z+1	TP_{z+1}	TS_{z+1}	TD_{z+1}
...	...	...	...
m	TP_m	TS_m	TD_m

Table 1: *Task_Table*

Task_Table is a brief description of the task to be executed. It contains the following information:

- *Task_ID* – a task identification number.
- *Task_Priority* – a task priority value.
- *Task_Specification* – contains a specific data related to the task.

  It can contain many fields depending and changing with task type, an example:

  target coordinates and number of robots required for the task.
- *Task_Done* – indicates that *Task_ID* is done.

4.3.2 *Bid_Table* description

The *Bid_Table* structure is shown on the below figure.

Robot_ID	Bid_Value	From	Signal_Strength	Done
1	BV_1	F_1	SS_1	D_1
2	BV_2	F_2	SS_2	D_2
...	...	...	...	...
c-1	BV_{c-1}	F_{c-1}	SS_{c-1}	D_{c-1}
c	BV_c	F_c	SS_c	D_c
c+1	BV_{c+1}	F_{c+1}	SS_{c+1}	D_{c+1}
...	...	...	...	...
n	BV_n	F_n	SS_n	D_n

Table 2: *Bid_Table*

The columns in the table are following:

- *Robot_ID* – a predefined unique identifier for a robot.
- *Bid_Value* – this is the bid value for the actual bidding. Potential ability of a robot to execute a task. This value is calculated locally by a robot based on data from *Task_Specification* and its own parameters (an example: actual position of robot and resource usage). Zero is a special value, it is a default/initial value and also it can inform that robot has communication failures.
- *From* – information from which robot the table was received. This is required to send back the fully filled table from the last robot of the Round Robin route to the first one.
- *Signal_Strength* – this is broadcast signal strength from robot which sent the table. This parameter is used during the second turn of Round Robin. It is useful for optimization the energy consumption and speed of transmission.
- *Done* – default zero, one when the robot has accomplished the task.

4.4 Exceptions to Protocol flow

In this section DAP will be analyzed to identify exceptions, which may occur during the seamless protocol flow and to show the solutions to handle them.

Quick identification of these exceptions and solving them increases robustness of the system. There are following types of exceptions: communication failures, robots' hardware malfunctions and decision issues after sorting tables: *Task_Table* and *Bid_Table*. These categories will be discussed below.

The basic strategies used to improve robustness of DAP are:

- monitoring of communication connectivity among robots by requesting acknowledges
- distributed repeated bidding which allows re-allocate task from failed robots to their ream members with the best announced potential to fulfill the task (representing by *Bid_Value*)

Below there are described reasons of failures and how DAP can recover from these failures by slightly degrading performance and maximizing the efficiency with resources which remain available to complete the task.

4.4.1 Communication failures

Below the communication failures which may occur during the two routes of Round Robin are discussed. Round Robin method is the way of information passing among the members of a robot team.

4.4.1.1 First Round Robin Route

- The initiator sends a *Task_Request* signal and waits for an acknowledgement from the first robot in the range. If there is no response within *waiting_for_aknowlegment_time* the initiator increases the signal strength by one step and waits again for acknowledgment. If it reaches the maximum strength (which by default covers the total area) that means that there are no robots available. So the task allocation is terminated.
- When robot R_c is increasing the range of the signal strength and in this range there is a failed robot then it is obviously omitted, because does not send an acknowledgment. So the robot R_c increases the signal strength to identify the next functioning robot R_{c+1}. The last robot in Round Robin Route increases its strength to the maximum covering total area then it assumes that robots not contained in the *Bid_Table* are in the state of communication failure. The solution is to not take them in the bidding, because there is

no communication with them. Instead the information which was originally to be delivered to robot R_{C+1} is sent to next robot $R_{C+2.}$

- Another failure occurs when number of detected robots is lower than the number of required robots for task execution. These robots will execute task with hope that some robots can be reassigned from the other task when available. If such a robot is not found the task is reported to the initiator as partially done.
- Robot R_C won last bidding and it is in the awarded robots group for particular task. Suddenly a malfunction causes a communication failure and it is deaf. This is similar to the situation when even the robot is not awarded in the next bidding; it continues to execute a task. It has been excluded from the bidding by its colleagues, but it does not know it because of communication failure. However the robot is self-disciplined and it recognizes that did not receive a new edition of the *Bid_Table* in the next cycle. In that case robot stops execution of the task.

4.4.1.2 Return Round Robin Route

When the last robot receives the *Bid_Table* and fills it - then the fully filled table is sent back to all robots also using the Round Robin manner. The data contained in columns *From* and *Signal_Strength* are used for speeding up transmission. Point-to-point communication is used, where the robot sends information to a neighbor with a predefined unique identifier *Robot_ID*.

Here may appear a failure – during the return round of sending information one robot failed. A method to detect this is following. Robot R_{C+1} sends a signal to robot R_C and waits over *waiting_for_aknowlegment_time*. If the respond does not arrive within this time robot R_C is ignored - Figure 7.

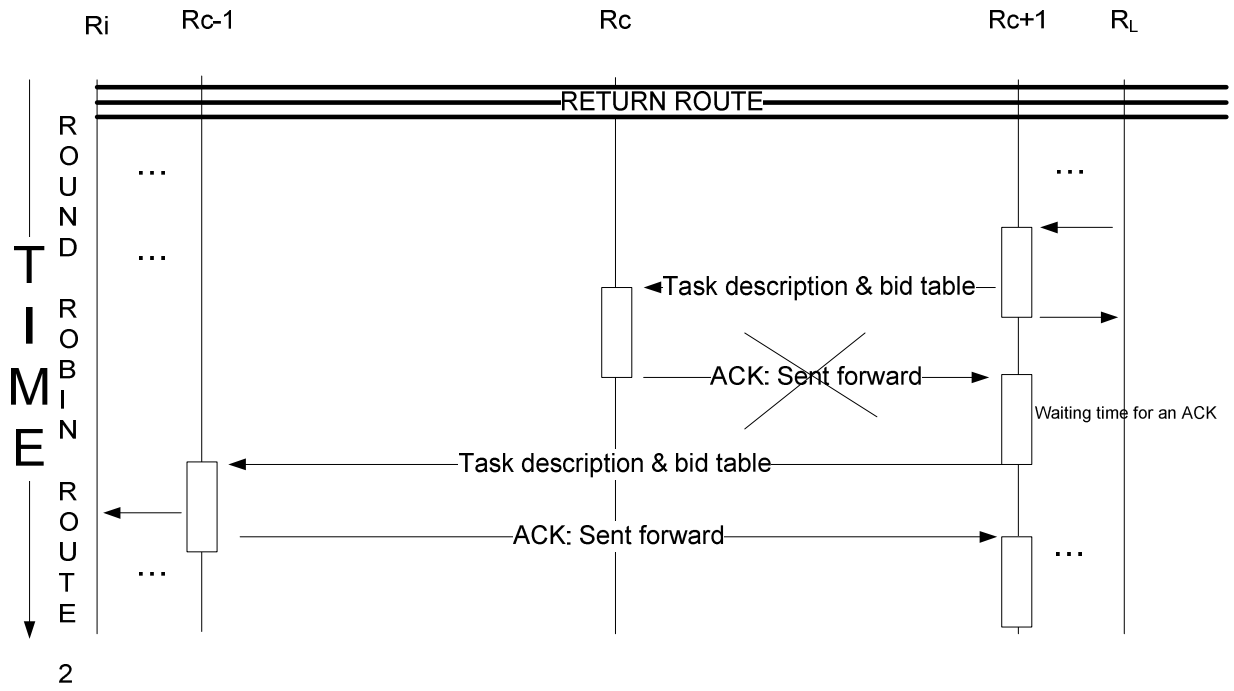


Figure 7: Communication failure during the Return Round Robin Route

Robot R_{c+1} writes zero to *Bid_Value* of robot R_c what means that R_c has a communication failure. This zero also eliminates robot R_c from the current bidding process (R_c has the lowest position on the bidding list). Some problems may occur with the previous robots with indexes from R_{c+2} to R_L. These robots do not know about the failure of R_c. This may lead to situation that these robots can award R_c in the current bidding and even one of them could be in the set of w awarded robots – its position is replaced undeservedly by robot R_c. However in the next biding the zero *Bid_Value* will be known to all robots from the beginning and such situation will not repeat.

It may also happen that in the next bidding the robot R_c may recover (will send acknowledgment) and set non-zero *Bid_Value* and can regain its position in award ranking.

To continue the current bidding robot R_{c+1} checks in the table from which robot R_c got a table before (R_{c-1}) and sends to it the message with the summed up *Signal_Strength.* If there is no acknowledgment again, the procedure is repeated until the initiating robot receives the fully filled table.

There is also possible a situation when during the second round of sending information one robot gets information, but failed and cannot transfer forward to the neighbor robot. This situation is detected by lack of acknowledgment within *waiting_for_aknowlegment_time*. This acknowledgment is sent only after forwarding the message by robot R_{c+1} to robot R_c.

4.4.2 Hardware malfunction failures

Hardware malfunction failures can be found in the following basic modules of a robot: perception, computation and execution.

The computational errors are described in the next section. The other failures can be divided into two categories – these which can be discovered by robot itself (like low power of battery) and these which cannot be discovered by robots during self-test.

Losing the battery power is easily monitored by robot itself. This is a contribution (one of parameters) to calculate a *Bid_Value*. And this way has an influence on the awarding and task execution. Different scenarios are possible how robot is to react after detecting different level of battery power. At which level it has to stop the task execution, at which level has to move to base–station for recharging. The interesting option can be on-the-run charging request which will be described in Future works chapter.

The example of the failure not detected by robot itself is that one of the robot's sensors gives wrong readouts. Based on them robot calculates *Bid_Value*, which could be better than the actual potential to fulfill the task. This value is reported to *Bid_Table* and the robot can be awarded to execute a task (even in the reality it should not).

It is difficult to find general solution to this type of failures as they are hardware related. However in the experimental part there is described an example of the broken GPS sensor.

The recovery from this failure is achieved by attentive monitoring by team members for the cost of more complicated algorithm and time consumption.

The example of execution module failure also presented in the experimental part is a situation when the robot is stuck when moving because of rough surface. The sensor and computational modules are working correctly – there are sending the signals to the execution module (motors) but there is no visible movement progress. Such failure is detected in deteriorating *Bid_Value* because there is no progress in target approach and also the battery level is going down. Such robot will be identified in a few cycles of DAP and de-allocated from the task. The best from the available robot will be assigned to this task as a replacement of the stuck robot. However if the stuck robot overcomes the difficult environment condition and starts to report better *Bid_Value* it can be again awarded to the previous or another task for which its *Bid_Value* is the best ranked.

4.4.3 Dealing with exceptions

In this section the decision issues are identified for different phases of protocol:

- Two robots in the same communication range
  In the next transmission step (only one robot can transmit at the moment, the others are in the listening mode) robot R_c has communicated with more than one robot in the same range. R_{c-1} has already sent *Bid_Table*, so it is in listening mode and does not send acknowledgment. Identified robots have indexes $R_{i,..,j}$. (Those indexes are not to be necessary sequential numbers). Arbitrary, the lower number is taken as an index for the next robot in the Round Robin Route.

- The same instances in *Bid_Table*
  All robots have received a fully filled *Bid_Table*. They sort the *Bid_Table* individually by column *Bid_Value*. If in the column *Bid_Value* are the same numerical values then the lower index robot is chosen. The same solution is applied respectively or *Task_Priority* – the highest *Task_ID* is then selected as the higher priority task.

- Not enough robots to execute a task
  Here may occur a problem that there are not enough robots to execute a task. In this situation the bidding is terminated and the task is waiting for more robots available, because high-priority tasks are allocated to the high-performance robots.

5 DAP – example of implementation

The implementation of the DAP protocol will be presented for the search and rescue multitask allocation problem.

5.1 Assumptions of an experiment

Robots have the below capabilities:

- Global Position System
- Radio transmitter-receiver, which changing range
- Surface of the area can change its characteristics

Resolution of *Bid_Value* is in the range [0..9] with the purpose to induce conflict situations and this way to observe a DAP behavior. Another reason is having a better visibility of simulation process.

5.2 Bid specification example

The bid parameter holds information about three robot features: resource usage, distance to target and estimated time to reach the target, all are summed up with different weights. Further it is represented as a column *Bid_Value*.

- Resource usage *(r)* – this is a status of both robots' batteries: logic unit and motors.
- Distance to target *(d)* – each robot knows its own geographical coordinates in the area. The task message carries information about the geographical coordinates of the target. Having these data each robot individually calculates the distance to target as a Euclidean distance between two points.
- Estimated time *(t)* – robot estimates the time from predefined data. It has a predefined map and two points: itself and the target. It calculates the optimal route with the inputs about surface type. Also the time required to make a next step may differ depending of type of area (surface) and the robot's capabilities. This parameter is very changeable, due to the dynamic environment.

An example of formula combining all these factors:

$$bid\_value \;=\; round\;(3 \bullet r + d + \frac{t}{2}) \;\in\; [0..9],\; where\; value \in N$$

Changing the formula and used arguments (being directly connected with robots resources) can better describe the real robots behaviors.

5.3 Simulated environment

The simulator is written with use of the MS.NET platform. In the main screen there are three panels: map of terrain, *Task_Table* and *Bid_Table.* At the right bottom (Figure 8) there is also a control panel where user can set the various modes of operation like:

- Round Robin signal visibility –signal propagation can be visible;
- Injection of new tasks;
- Simulate failure – here the user has an access to the *Task_Table* and *Bid_Table* and can change their contents to induce failures which can be further analyzed in the step-by-step mode;
- Step-by-step mode – after sort and allocation process the application stops;

  During the step-by-step working user can watch and change, when required all parameters in *Task_Table* and *Bid_Table* which influence on robots behavior.
- Snapshot mode – at any time tasks and robots displacement can be snapshot and saved as input scenario for further analysis.

Additionally from the upper menu bar the user has access to such variables like: number of robots and tasks; speed and size of robots; etc.

From this menu user can also load/save different maps and scenarios.

Predefined scenarios are used to simulate the failures and other exceptions to the main algorithm flow for detailed analyses – for instance in step-by step mode.

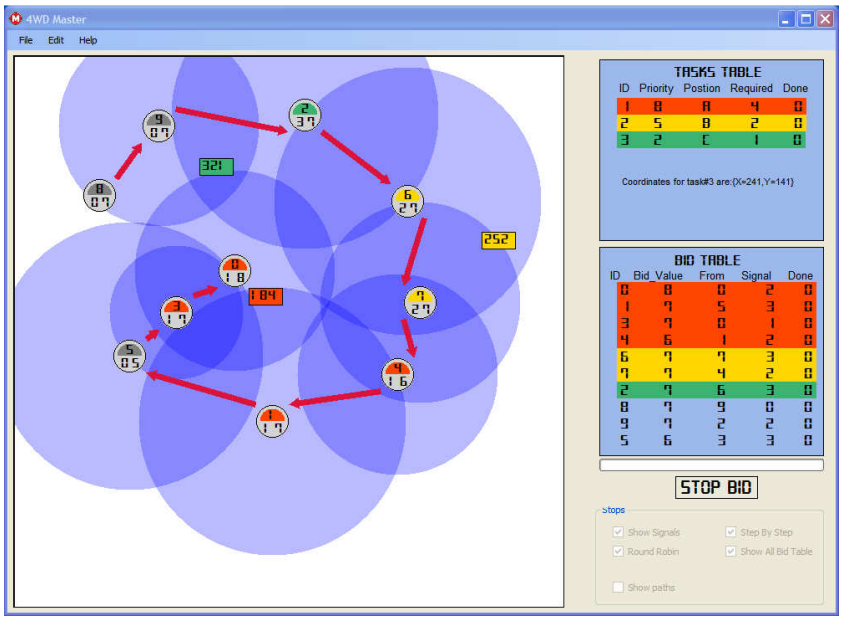


Figure 8: DAP simulator environment

On the screen robots are represented as gray circles (if are not allocated) with three digits: *Robot_ID*, *Bid_Value, Task_ID*. Depending on the allocated task the color in the upper part of the robot is changed. Robot is an omni-directional type that means that it does not have to spin the body to go to the target, it is always faced to north direction.

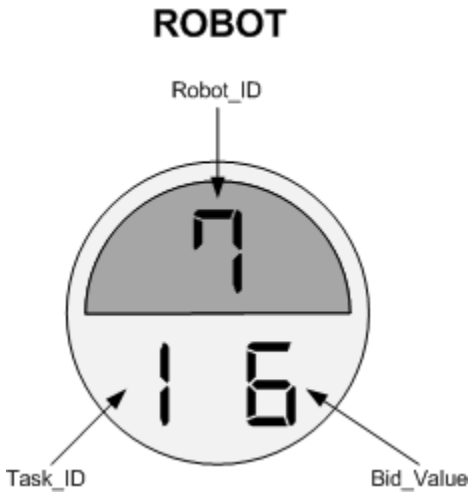


Figure 9: Robot design

The task is represented as a colorful rectangle (color depends on *Task_ID* parameter). It has also three digits: *Task_ID*, *Task_priority*, *required robots*. Task does not move itself. Nevertheless task coordinates can be changed by an operator in the step-by-step mode.

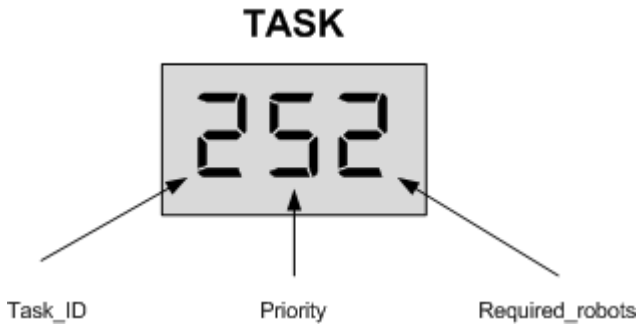


Figure 10: Task design

5.4 Algorithm steps

In the simulated environment tasks and robots displacement can be placed randomly or loaded from the earlier saved scenarios.

Bid_Value is normalized to be the integer number from 0 to 9. The range of these numbers was intentionally limited to be only one digit – it will make easier and nicer the visualization during simulation. However this limit should not have impact on general considerations.

The steps of the algorithm have been illustrated upon the below figures.

On each of them you can see what activity was done (e.g. emitting/receiving a *Bid_Table*). The aside table shows *Bid_Table* contents after such activity.

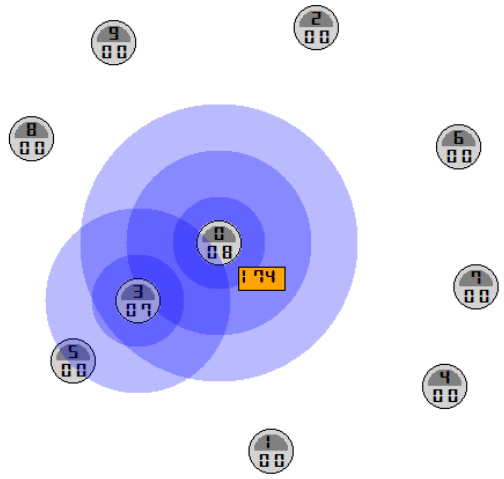


Figure 11: First Round Robin Route step 1

BID TABLE #3

ID	Bid_Value	From	Signal	Done
0	8	0	2	0
3	7	0	1	0

Table 3: *Bid_Table* of robot R_3

The first bid is initiated by robot R_0 which sends the table to the nearest robot (in this case R_3). Data in the table referring to robot R_0 – calculated *Bid_Value*= 8, received from robot R_0 (initialization). Upon initialization the table has all 0 values (not shown for better clarity). Robot R_3 inputs its *Bid_Value*=7 and indicates that the table was sent to it from robot R_0.

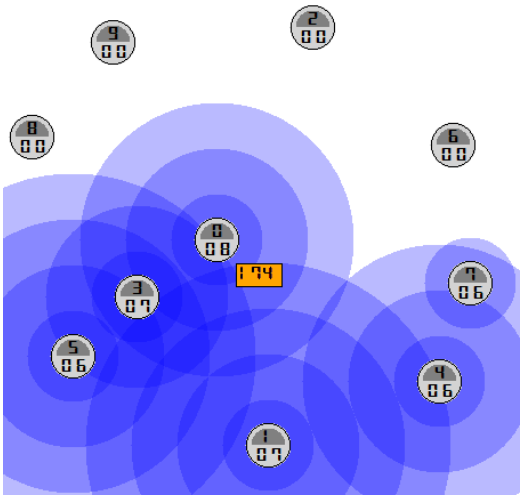


Figure 12: First Round Robin Route next steps

BID TABLE #7

ID	Bid_Value	From	Signal	Done
0	8	0	2	0
1	7	5	3	0
3	7	0	1	0
4	6	1	2	0
5	6	3	3	0
7	6	4	1	0

Table 4: *Bid_Table* of robot R_7

The above picture shows the signal propagation among robots after 6 steps of Round Robin. Robot R_c is increasing range of the signal step-by-step if in this range is another robot then they exchange information and change their states: R_c from sender mode becomes a deaf robot and R_{c+1} from listening mode enters into sender mode. The situation is repeated for other robots.

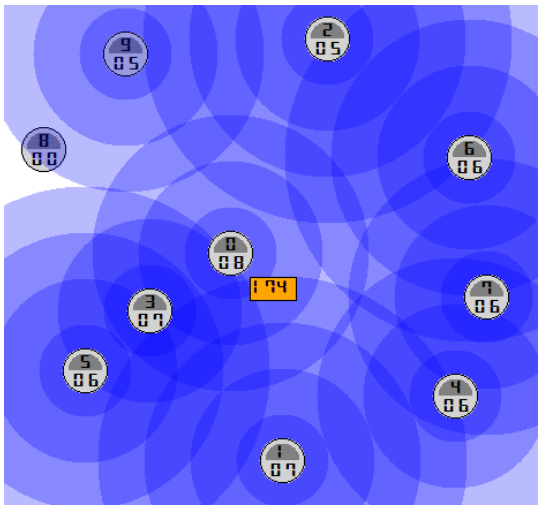


Figure 13: First Round Robin Route last step

BID TABLE #8

ID	Bid_Value	From	Signal	Done
0	8	0	2	0
1	7	5	3	0
2	5	6	3	0
3	7	0	1	0
4	6	1	2	0
5	6	3	3	0
6	6	7	3	0
7	6	4	2	0
8	0	9	0	0
9	5	2	2	0

Table 5: *Bid_Table* of robot R_8

Based on the algorithm of Round Robin all robots get the table, but the fully filled is in the possession of only the last involved robot R_8.

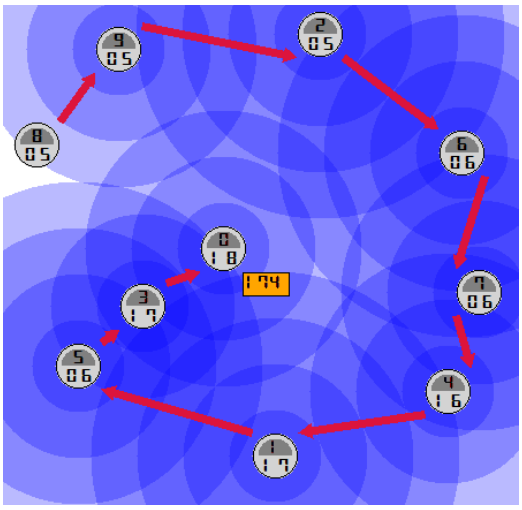


Figure 14: Return Round Robin Route

BID TABLE

ID	Bid_Value	From	Signal	Done
0	8	0	2	0
1	7	5	3	0
2	5	6	3	0
3	7	0	1	0
4	6	1	2	0
5	6	3	3	0
6	6	7	3	0
7	6	4	2	0
8	5	9	0	0
9	5	2	2	0

Table 6: Fully filled *Bid_Table* in memory of each robot after sorting

The fully filled table is to be sent to back to all robots also using the Round Robin algorithm. For sending the data contained in column *From* is used. The arrows show the path of the signal during the second Round Robin route.

Now all robots have received a fully filled table. They sort table (Figure 7) individually by column *Bid_Value*. If in this column are more than one the same value than they sort by *ID* (lower is better).

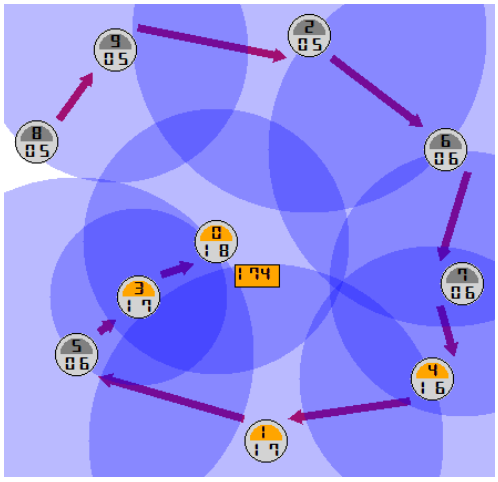


Figure 15: Robots allocated to the task

BID TABLE

ID	Bid_Value	From	Signal	Done
0	8	0	2	0
1	7	5	3	0
3	7	0	1	0
4	6	1	2	0
5	6	3	3	0
6	6	7	3	0
7	6	4	2	0
2	5	6	3	0
8	5	9	0	0
9	5	2	2	0

Table 7: Tasks allocation across robots

They individually check if their number *ID* is included in top four positions of the table. If yes they proceed with task execution, otherwise they stop and enter into the listening mode.

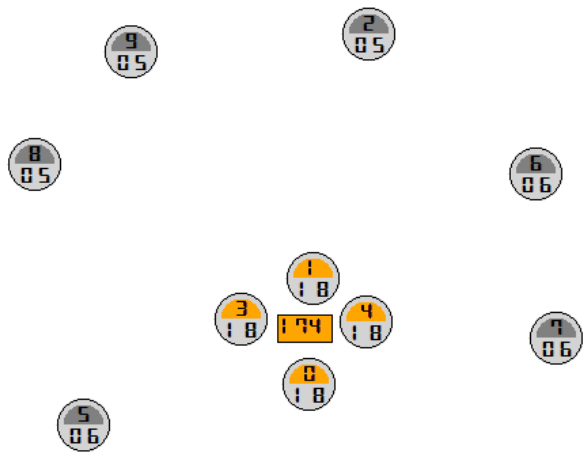


Figure 16: Robots accomplished a task

BID TABLE

ID	Bid_Value	From	Signal	Done
0	8	0	0	1
1	8	0	0	1
3	8	0	0	1
4	8	8	4	1
5	6	7	2	0
6	6	2	2	0
7	6	6	5	0
2	5	4	3	0
8	5	9	3	0
9	5	0	2	0

Table 8: *Bid_Table* after task accomplishment

The top four robots are awarded in this bidding. These are robots numbered 0, 1, 3 and 4. They continue to execute the job - in this case approaching the target. If robot reaches the target it checks the column *Done*. If all required for the task robots *Done* that means that the task was accomplished

For the multitasking the basic mechanism is the same with the modification that now the *Bid_Value* is stored in the one-dimensional table where each position corresponds to the each task.

Table 9 is the illustration of *Task_Table for* three tasks.

The *Task_Table* is sorted by priority. Robot takes the highest priority *Task_ID* and sorts *Bid_Table* related to first task and allocates required number of robots (4). Then the start index of sorting *Bid_Table* is increased to the sum of required robots from last allocated tasks so now is starts from 4. Robot takes another task with lower priority and refreshes *Bid_Value* column. Then the required robot for task *ID*=2 is 2. Robot sorts *Bid_Table* from start index 6. Then available robots are allocated to tasks of lower priority. The sorting process is finished when all tasks have allocated robots.

TASK_TABLE				
Task_ID	*Task_Priority*	*Coordinates*	*Required_Robots*	*Task_Done*
1	7	TC_1	4	0
2	5	TC_2	2	0
3	2	TC_3	1	0

Table 9: *Task_Table* for 3 tasks

Task allocation

Figure 17 shows that there are 3 tasks injected. Robots calculate *Bid_Value* for each task. Now the *Bid_Value* is an array of three elements. This information spreads to all the robots via Round Robin. After the second route of Round Robin each robot has the same Table 10.

Then a first column of *Bid_Value* corresponding to the task T1 is sorted. The number of required robots is 4, so the best four robots are awarded: R_0, R_1, R_3, R_4 (Table 11).

For the task T2 there are only 2 robots required. They are chosen from the second column of *Bid_Value* (corresponding to task T2) after sorting. Sorting is done only for robots not yet allocated. Robot R_8 and R_9 are allocated for the task T2.

For the task T3 there is only one robot required. After the sorting of only unallocated robots the robot R_7 is chosen. (Figure 20, Table 13).

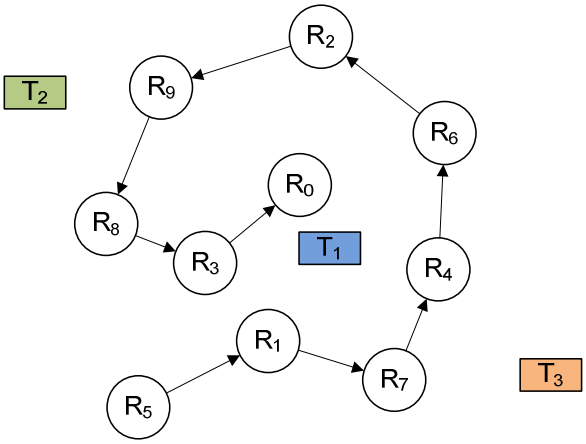


Figure 17: Round Robin Route for 3 tasks

ID	*Bid_Value*			*From*	*Signal*	*Done*
0	9	5	6	0	3	0
1	9	3	6	7	3	0
2	4	4	5	9	6	0
3	8	6	4	0	1	0
4	8	4	8	6	2	0
5	2	2	2	1	6	0
6	5	3	7	5	0	0
7	5	1	9	4	3	0
8	4	9	4	3	2	0
9	3	9	3	8	2	0

Table 10: Updated *Bid_Table* for 3 tasks

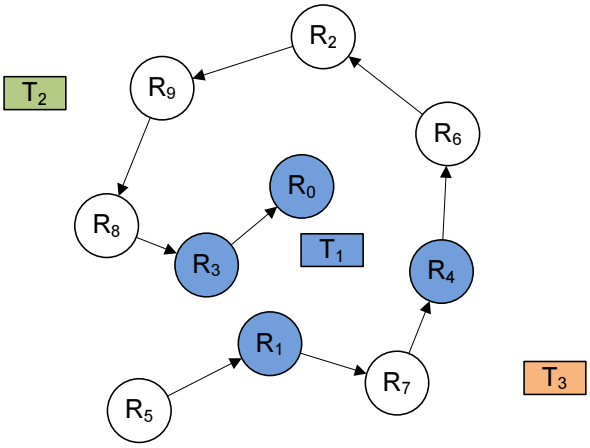


Figure 18: Robots allocated to the first task

ID	*Bid_Value*			*From*	*Signal*	*Done*
0	9	5	6	0	3	0
1	9	3	6	7	3	0
3	8	6	4	0	1	0
4	8	4	8	6	2	0
6	5	3	7	5	0	0
7	5	1	9	4	3	0
2	4	4	5	9	6	0
8	4	9	4	3	2	0
9	3	9	3	8	2	0
5	2	2	2	1	6	0

Table 11: *Bid_Table* sorted by first task

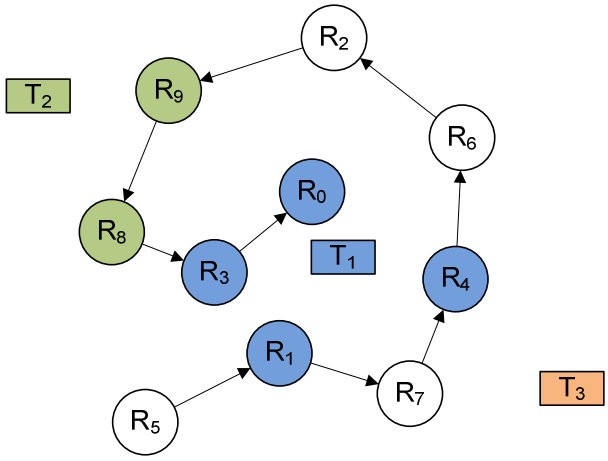


Figure 19: Robots allocated to the second task

ID	*Bid_Value*			*From*	*Signal*	*Done*
0	9	5	6	0	3	0
1	9	3	6	7	3	0
3	8	6	4	0	1	0
4	8	4	8	6	2	0
8	4	9	4	3	2	0
9	3	9	3	8	2	0
6	5	3	7	5	0	0
7	5	1	9	4	3	0
2	4	4	5	9	6	0
5	2	2	2	1	6	0

Table 12: *Bid_Table* sorted by second task

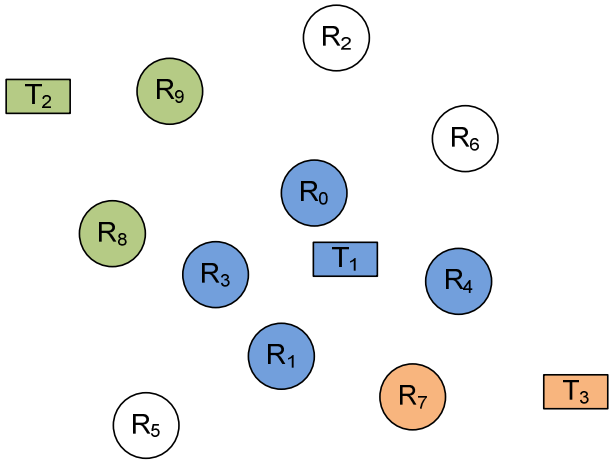


Figure 20: Robots allocated to the third task

ID	*Bid_Value*			*From*	*Signal*	*Done*
0	9	5	6	0	3	0
1	9	3	6	7	3	0
3	8	6	4	0	1	0
4	8	4	8	6	2	0
8	4	9	4	3	2	0
9	3	9	3	8	2	0
7	5	1	9	4	3	0
6	5	3	7	5	0	0
2	4	4	5	9	6	0
5	2	2	2	1	6	0

Table 13: *Bid_Table* sorted by third task

5.5 Dynamic environment

The above description was for a uniform terrain, but real surfaces may change from point to point. During robots movement they may encounter different types of surface. This can cause problems with moving (slow down or even stuck). They can also loose battery power while avoiding unexpected obstacles. Figure 21 presents an example from the real the world, in this case a tropical island.

In this version of the implementation robots change speed and resource usage depending on the surface type. They can drive or sail on the water. However passing through different types of surface has an impact on speed and battery usage what is reported in *Bid_Table*.

Robots in simulator detect the type of surface depending on color and based on this the actual speed is changed, which has a huge contribution to the resource usage (a factor in *Bid_Value* equitation). In the situation when speed is low and power consumption big, then the *Bid_Value* of such robot deteriorates and it can be possible that other robot in the next Round Robin route will replace this slower robot.

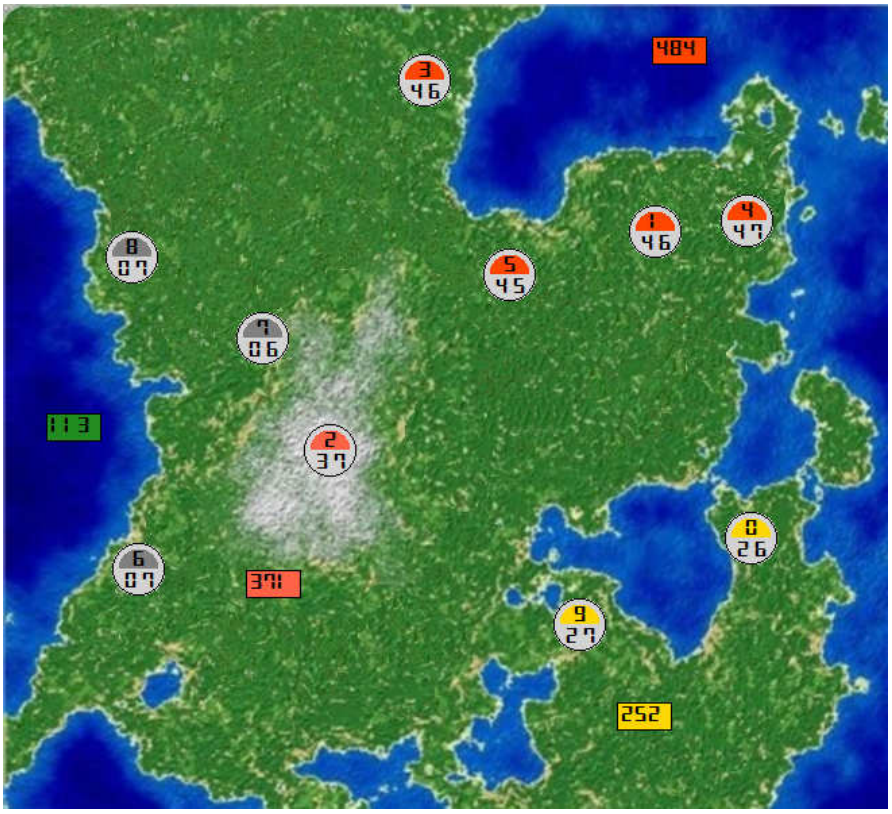


Figure 21: Robots in dynamic changed environment

6 Evaluation

DAP was evaluated through multiple experiments done in the multi-robot simulation system. The below described scenarios were tested to check the main algorithm idea, simulate failures and observe how DAP recovers from them.

There were simulated failures of robots and DAP identified the failed robots and eliminated them from the task execution. Since none of the robots was playing a key role, the task execution was continued. Such a robot was excluded from the task and replaced by another robot. If the failure of robot was temporary (e.g. communication noise) and the robot can recover it can again participate in the bidding/awarding process. DAP can recover from these failures by slightly degrading performance and maximizing the efficiency with resources which remain available to complete the task.

The following failures were induced in the simulator:

Communication failures

- During the First Round Robin Route one robot is failed.
- During the Return Round Robin Route one robot is failed
- The above failures were repeated with more involved failed robots
- During the Return Round Robin Route one robot gets a message, but fails and cannot send forward.
- Temporary loss of communication

  .

Hardware malfunction failures

- Robot has a broken sensor, its GPS works improperly.
- Low battery level
- Robot stucks when moving because of rough surface

Dealing with exceptions

- More than one robot in the same communication range
- More than one task with the same *Task_Priotity*
- More than one the same *Bid_Value* in the *Bid_Table*
- Number of available robots lower than *required_robots* for a task

Communication failures

- During the first Round Robin Route one robot is failed.

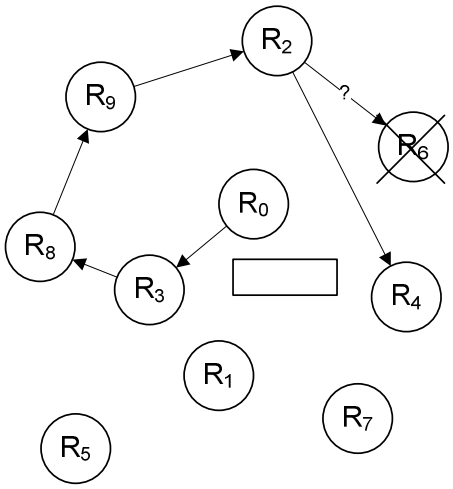


Figure 22: Communication failure during the First Round Robin Route - detection

ID	*Bid_Value*	*From*	*Signal*	*Done*
0	3	0	3	0
1				
2	8	9	6	0
3	4	0	1	0
4				
5				
6				
7				
8	6	3	2	0
9	7	8	2	0

Table 14: *Bid_Table* of robot R_2

Robot R_2 wants to send information to the nearest robot. Robot R_6 which has communication failure and does not respond with acknowledgment, so R_6 re-sends message to R_4.

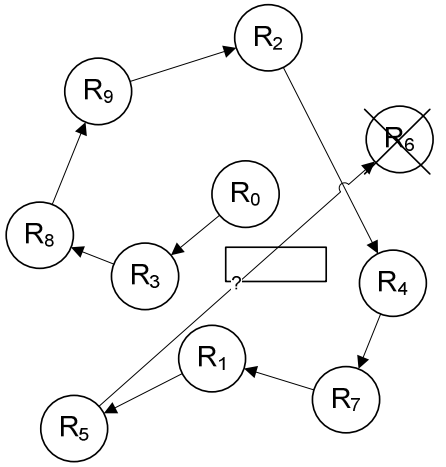


Figure 23: Communication failure during the First Round Robin Route - repair

ID	*Bid_Value*	*From*	*Signal*	*Done*
0	3	0	3	0
1	6	7	3	0
2	8	9	6	0
3	4	0	1	0
4	6	6	2	0
5	9	1	9	0
6	0			
7	8	4	3	0
8	6	3	2	0
9	7	8	2	0

Table 15: *Bid_Table* of robot R_5

R_5 as the last robot in the Round Robin sees that there is no data for R_6 filled in the *Bid_Table*. So it tries to communicate with it by increasing the strength of the signal. If it reaches the maximal strength of signal and there is no acknowledgment from R_6 so R_6 is supposed to have a failure. To indicate this situation R_5 writes to the table *Bid_Value*=0 (which is a special value) and *From*=5. So R_6 is not available in the current bidding, but can participate in next bidding if the environment is changed or its communication perception is recovered.

- During the return Round Robin route one robot is failed

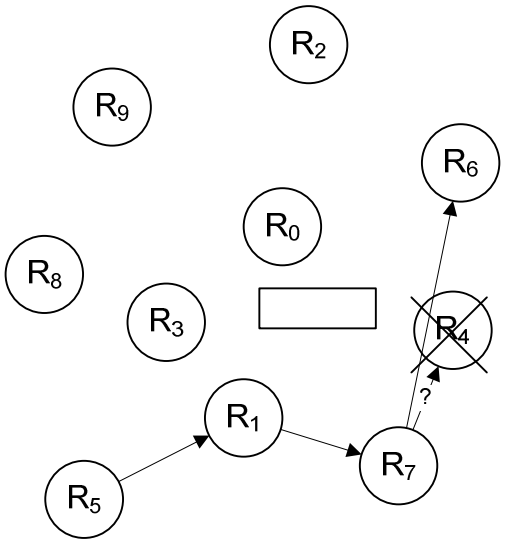


Figure 24: Communication failure during the Return Round Robin Route - detection

ID	*Bid_Value*	*From*	*Signal*	*Done*
0	3	0	3	0
3	4	0	3	0
1	6	7	6	0
4	0	7	1	0
8	6	3	2	0
9	7	8	9	0
2	8	9	3	0
6	8	2	3	0
7	8	4	2	0
5	9	1	2	0

Table 16: Sorted *Bid_Table* of robot R_7

During the return Round Robin route R_7 wants to send to R_4, because it has in the column *From* that it should send there. If there is no acknowledgement within *waiting_for_aknowlegment_time* R_6 sends to R_4 because in the column *From* there is an indication that robot R_4 was preceded by robot R_6.

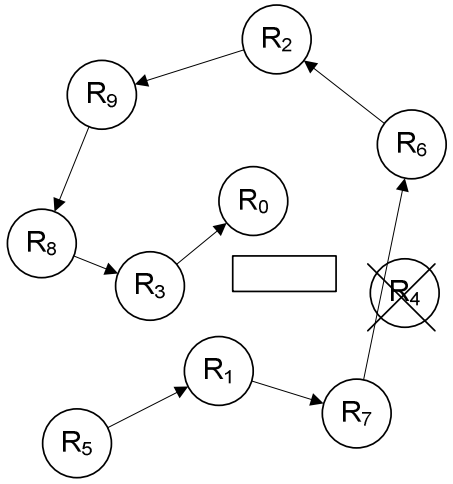


Figure 25: Communication failure during the Return Round Robin Route - repair

ID	*Bid_Value*	*From*	*Signal*	*Done*
0	3	0	3	0
3	4	0	3	0
1	6	7	6	0
4	0	0	0	0
8	6	3	2	0
9	7	8	9	0
2	8	9	3	0
6	8	2	3	0
7	8	6	2	0
5	9	1	2	0

Table 17: Sorted *Bid_Table* of all robots without robot R_4

During the next bidding R_4 does not participate. The consequence is that R_7 writes R_6 instead of R_4 to column *From.* R_1 and R_5 don't know that R_4 has failed, but in the next bidding the mechanism described above will work and information that R_4 is not available in this moment will be communicated to all robots.

- During the return Round Robin route one robot gets a message, but fails and cannot send forward.

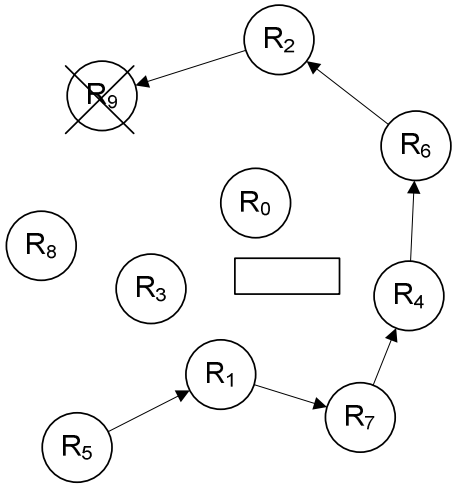


Figure 26: Communication failure: receive-yes, send-no detection

ID	*Bid_Value*	*From*	*Signal*	*Done*
0	3	0	3	0
3	4	0	3	0
1	6	7	6	0
4	6	6	1	0
8	7	3	2	0
9	7	8	9	0
2	8	9	3	0
6	8	2	3	0
7	8	4	2	0
5	9	1	2	0

Table 18: *Bid_Table* of the failed robot R_9

During the second round of sending information R_9 gets a message, but fails and cannot send forward. The protocol is constructed that after sending the table, the back acknowledgment is sent to the robot which delivered this table. If after *waiting_for_aknowlegment_time* an acknowledgement has not arrived then R_2 looks up in the table the column *From* to send table to another robot (in this case R_8).

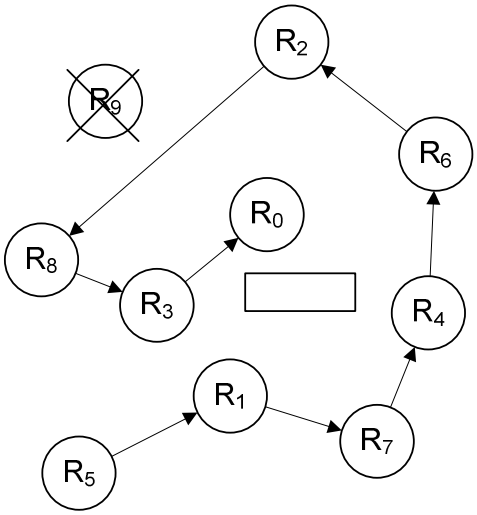


Figure 27: Communication failure: receive-yes, send-no detection repair

ID	*Bid_Value*	*From*	*Signal*	*Done*
0	3	0	3	0
1	6	5	3	0
2	8	6	6	0
3	4	0	1	0
4	6	7	2	0
5	9	8	9	0
6	8	4	3	0
7	8	1	3	0
8	7	3	2	0
9	0	0	2	0

Table 19: *Bid_Table* of the robot R_9

During the return Round Robin route R_9 is not bidding. However communication may recover and it can participate in the next bidding. The failure is marked by *Bid_Value*=0, which will guarantee that it will not be in the team of the best robots; it is eliminated from the task.

- Robot did not have the connection but move to the target; when it appears near the target the communication recovers and it takes part in next bidding.

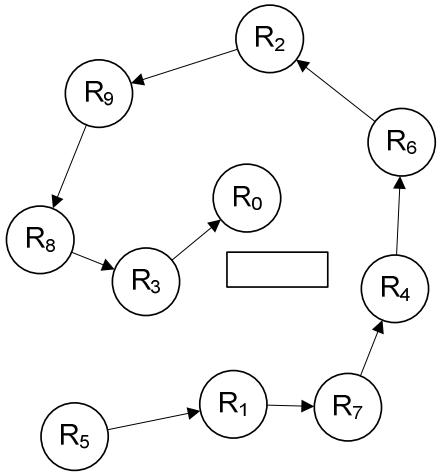
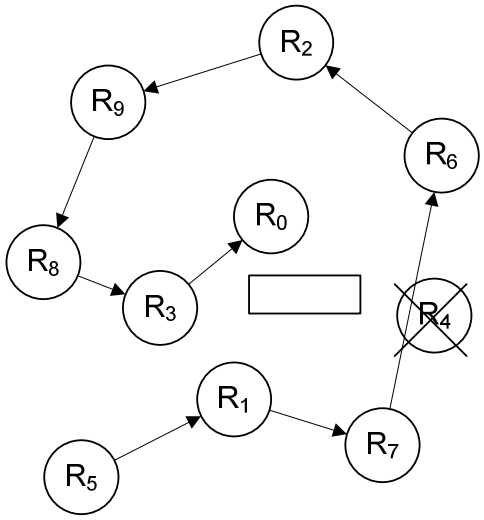


Figure 28: Loss of communication - detection

ID	*Bid_Value*	*From*	*Signal*	*Done*
0	9	0	3	1
3	8	0	3	0
1	7	7	6	0
4	6	6	1	0
7	5	4	2	0
6	4	2	9	0
9	4	8	3	0
2	3	9	3	0
8	2	3	2	0
5	1	1	2	0

Table 20: *Bid_Table* for loss of communication - detection

R_4 was in the top of the table, but suddenly lost communication capabilities.



ID	*Bid_Value*	*From*	*Signal*	*Done*
0	9	0	3	1
3	8	0	3	1
1	7	7	6	0
7	5	4	1	0
6	4	2	2	0
9	4	8	9	0
2	3	9	3	0
8	2	3	3	0
5	1	1	2	0
4	0	7	2	0

Table 21: *Bid_Table* of the robot R_7

Figure 29: Loss of communication - repair

R_4 does not know that next bidding does not include it. It has a communication failure, but good motors and GPS.

After the *contract_exeuction_time* robot R_4 stops the execution and waits for the next bidding. But in this case, robot R_4 has a communication failure and will not get any more information from others robots. Because the rest of robots assume that R_4 has a failure (what is marked with *Bid_Value* =0. So R_4 is de-allocated from the task and R_4 with the next highest will replace R_4.

- Robot has a broken sensor, its GPS works improperly.

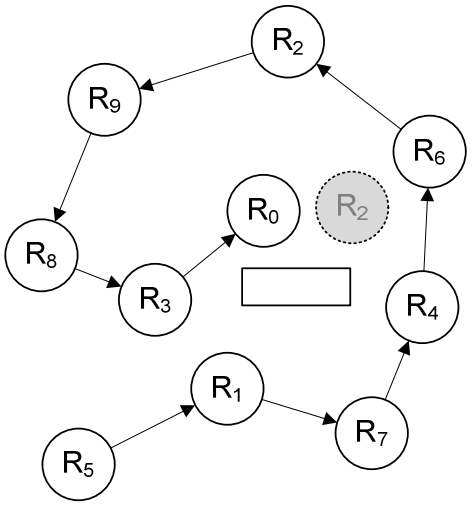


Figure 30: GPS failure - detection

ID	*Bid_Value*	*From*	*Signal*	*Done*
0	9	0	3	1
3	8	0	3	0
2	8	7	6	0
1	7	6	1	0
4	5	4	2	0
6	4	2	9	0
9	4	8	3	0
7	3	9	3	0
8	2	3	2	0
5	1	1	2	0

Table 22: *Bid_Table* for loss of communication - repair

R_2 is in a group on the awarded robots, even it is far from the target in reality because due to GPS offset it calculates its *Bid_Value=8*, in reality it should be *Bid_Value*=4. The real position is drawn as a regular circle, while the reported position is shadowed with grey.

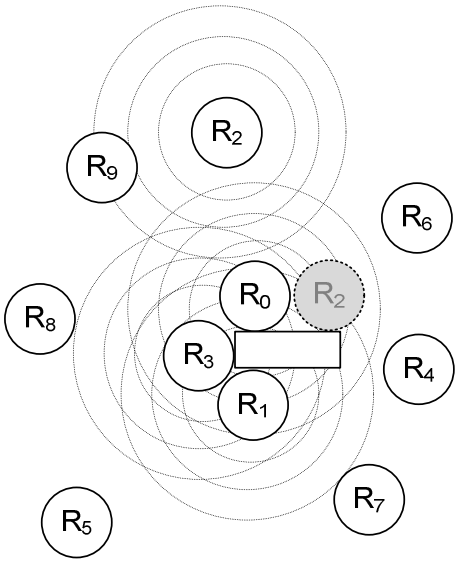


Figure 31: GPS failure - repair

ID	*Bid_Value*	*From*	*Signal*	*Done*
0	9	0	3	1
3	9	0	3	1
1	9	7	3	1
2	9	4	3	1
4	6	2	0	0
9	4	8	0	0
6	3	9	0	0
8	2	3	0	0
5	1	1	0	0
7	1	7	0	0

Table 23: *Bid_Table* of the failed robot R_2

The identification of this failure can be done when all other robots allocated for the task will reach a target. If there are at least two robots reporting that the task is done then their physical neighborhood is to be checked. This is done by sending signal of the Strength =3. In the described situation it will be discovered that R_2 in fact has not reached the target. So it is to be de-allocated from the task and in the next binding procedure the best available robot will be assigned to this task – in this case robot R_4.

The experiments performed in the simulated environment confirmed the main idea of DAP algorithm. Tasks were allocated according to priorities with cost minimization - time to execute a task with minimal resource (battery) usage. DAP proved also that can handle multi task injection.

There were different types of failures (communication failures, partial hardware malfunctions) simulated in various scenarios. DAP proved that can recover from these failures by locating and eliminating the failed robot and then finding the best replacement for it.

The failures simulation and DAP recovery from failures proved the DAP robustness.

Also the DAP confirmed the flexibility to environment changes causing that bid values of robots were deteriorated due to surface conditions. In such situation DAP was still able to choose the best robots from all available to execute a task.

7 Conclusions and future works

Conclusion

In this thesis a Distributed Award Protocol (DAP) was presented which is a modified Contract Net Protocol, where the main modification is in the bidding process.

The novel approach is based on a bidding/awarding mechanism without a distinguished manager. Robots communicate with each other by passing information in a Round Robin manner. As a result all of them are in the possession of global knowledge about injected tasks and bid values of all robots. This contributes to assign tasks to robots in the optimal way with the aim to minimize a global cost function which is a compromise between cost of task execution and cost of resources usage.

DAP uses a distributed negotiation process with no manager involved. This implies the following benefits: no communication bottleneck and low computational power requirement. Not having a manager eliminates a sensitive point of robustness: if the manager fails the all team robots are inoperable. However the team members may also fail. DAP can handle such situations through Round Robin information passing among robots.

DAP can identify a failed robot, de-allocate it from the task and find the best suited robot to allocate it to the task as a replacement and thus continue task execution by a team.

DAP can accept new tasks dynamically and reallocate tasks with the aim to minimize a global cost function. High-priority tasks are allocated to the high-performance robots.

The robots can be reassigned from one task to another after each bidding process. This is due to different bid values (in the presented example it was battery power drop and slow moving towards target due to tough surface type). This features shows flexibility to environment changes.

The next reason for the changing of robots assignment is the appearance of a new task with higher priority which can cause a robot almost approaching target in the current task to be delegated to the higher priority task. This gives a flexible reaction to dynamically injected tasks to re-assign tasks with aim to optimize costs and resource usage.

The features of DAP are summarized in the below table.

Feature	Advantage	Drawback
Distributed allocation mechanisms	Lack a single point of failure, so more robust to failures of robots	
No manager/leader (Act local)	No communication bottleneck	Decisions made by group, consensus required
Each robot knows all tasks and bid values of all robots (Think global)	Optimization of tasks assignment to minimize a global cost function	Too much information sent to team members
Distributed negotiations	No need for big computational power	–
Frequent bidding in Round Robin manner	• Increases robustness • Reallocates tasks among robots more efficiently • Resource/cost optimization	Time/energy consuming
Tasks dynamically injected/allocated	Flexible to dynamical environment	–
Frequent auctioning and bidding	Flexibility to the task priority changes	–

Table 24: DAP features

Future work

Future work is planned including simulator improvements and also the testing of DAP in real world.

Simulator improvements

Regarding the simulator the following improvements have been already identified:

- Tasks groups conversation

  The purpose is to improve the performance of passing information. Only awarded robots to the particular task talk to each other – a Round Robin is limited only to the engaged robots. When one of them fails then they start to make a new route of Round Robin including all remaining robots this time. The aim is to get a new robot for this task as a replacement for the failed robot.

- Introduction of heterogonous robots

  Each robot has different sensors and capabilities dedicated to selected tasks only. Heterogeneity robots have different task execution performance- e.g. different speed. To execute a complex task a squad of selected robots will be required.

- Introduction of complex task allocation

  Now only simple tasks are implemented, which can be executed by a robot in a straightforward, prescriptive manner. The future work will focus on complex tasks that can be decomposed into multiple inter-related subtasks

- Introduction of special purpose charging robot

  This is a proposal to introduce special mobile robot which charging capabilities as an alternative for stationery charging-station. Such robot would be called to arrive to the robot busy with the task execution to recharge its battery on-the-run instead of leaving a task and going to base station to recharge.

Real world testing

Regarding experiments in the real world it is planned to use robots based on the 4WD platform. 4WD[1] was constructed for creating robots formations, but with small modifications it can be successfully implemented in DAP.

The indication to use mobile robots for testing is a feature of DAP that thanks to its distributed architecture there is no request for high computational power so embedded systems used in mobile robots will ideally fit for this purpose.

Figure 32: 4WD photo

4WD robots use 4 wheel drive and can rapidly change the trajectory at any angle. They have distance and proximity sensors round the body to avoid collisions. This enables to check DAP in the situation when robot trying to avoid collision changes trajectory and hence its *Bid_Value*. (This situation is not included in software simulator).

[1] 4WD was awarded the 2nd prize at mobile robot competition Cybertech 2013 UPM

8 Glossary

Initiator	-	An agent requesting a task execution
Bidding	-	A process where the best robots are awarded to execute a task
Awarded robot	-	A robot which won the bidding
Robot	-	A physical agent
Round Robin Route	-	A method of information passing among robots
Target	-	A destination in an example, which robots should reach
Task	-	An abstract to execute
Task allocation	-	A method to allocate robots to required tasks
Global cost function	-	A complex function calculating a total cost of task execution
Award-based protocol	-	A type of protocol where the best robots are allocated to execute a task
Message	-	A packet of information sent among robots via Round Robin manner
Signal range	-	Broadcast signal range increase in steps to preserve power
Acknowledgment	-	A confirmation of receiving a message

Bibliography

[1] Aaron, G., & Murphy, R. (1999). Affective Recruitment of Distributed Heterogeneous Agents. *Proceeding of the National Conference on Artificial Intelligence*, (pp. 14-19). London.

[2] Antidio, V., Maza, I., & Ollero, A. (2007). SET: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets. *IEEE International Conference on Robotics and Automation*, (pp. 3339-3344). Roma, Italy.

[3] Botelho, S., & Alami, R. (1999). M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. *IEEE International Conference on Robotics and Automation*, (pp. 1234-1239). Detrit.

[4] Bourne, B., & Jennings, N. (2001). Reasoning about Commitments and Penalties for Coordination between Autonomous Agents. *Proceeding of the 5th International Conference on Autonomous Agents* .

[5] Campbell, A., & Wu, A. (2009). On the significance of synchroneity in emergent systems. *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent System*, (pp. 449-456).

[6] Campbell, A., & Wu, A. S. (2011). Multi-agent role allocation: issues, approaches, and multiple perspective. *Aution Agent Multi-Agent System* , 317-355.

[7] Cao, H., Lacroix, S., Ingrand, F., & Alami, R. (2010). Complex Task Allocation for Multi Robot Teams under Communication Constraints. *5th National Conference on "Control Architecture of Robots".* Douai.

[8] Chaimowicz, L., Campos, M., & Kumar, V. (2002). Dynamic Role Assignmnet for cooperative Robots. *Proceeding of the IEEE Intenrational Conference on Robotics and Automation*, (pp. 293-298). Pennsylvania.

[9] Choi, H.-L., Brunet, L., & How, J. P. (2009). Consensus-Based Decetralized Auction for Robust Task Allocation. *IEEE Transactions on Robotics* , 912-926.

[10] Dasgupta, P. (2009). *A Dynamic-bid Auction Algorithm for Cooperative, Distributed Multi-Raobot Task Allocation.* UNO Computer Science Technical Report.

[11] Davis, R., & Smith, R. G. (1983). Negotiation as a Metaphor for Distributed Problem Solving. *Artificail Intelligence* , 63-109.

[12] Dias, B. M. (2004). *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments.* Pittsburgh, Pennsylvania: The Robotics Institue Carnegie Mellon University.

[13] Dias, B. M., Zlot, R., Kalra, N., & Stentz, A. (2006). Market-Based Multirobot Coordination: A Survey and Analysi. *Proceeding of the IEEE - Special Issue on Multi-Robot Systems* , 15-31.

[14] Dias, B. M., Zlot, R., Zinck, M., Gonzalez, J. P., & Stentz, A. (2004). A Versatile Implementation of the TraderBots Approach for Multirobot Coordiantion.

[15] Distributed Intelligence: Overview of the Field and its Application in Multi-Robot Systems*Journal of Physical Agents* 5-14

[16] Doniec, A., Bouraqadi, N., & Defoort, M. (2009). Distributed constraint reasoning applied to multi-robot exploration. *21st IEEE International Conference on Tools with Artificial Intelligence*, (pp. 159-166).

[17] Douglas, V., & Veloso, M. (2009). Multi-robot dynamic role assignment and coordination. *Multi-Robot Systems* , 87-98.

[18] Foundation for Intelligent Physical Agents. (2002). *FIPA Contract NET Interaction Protocol Specification.* Geneva.

[19] Gerkey, B. P. (2003). *On multi-robot task allocation.* Los Angeles: University of Southern California.

[20] Gerkey, B. P., & Matarić, M. J. (2004). A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research* , 939-954.

[21] Gerkey, B. P., & Matarić, M. J. (2001). Principled Communication for Dynamic Multi-Robot Task Allocation. *Experimental Robotics* , 353-362.

[22] Gerkey, B. P., & Matarić, M. J. (2002). Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination. *Proceedings of the IEEE International Conference on Robotics and Automation* , 464-469.

[23] Gerkey, B. P., & Matarić, M. J. (2002). Sold!: Auction Methods for Multirobot Coordination. *IEEE Transaction on Robotics and Automation* , 758-768.

[24] Hunsberger, L., & Grosz, B. (2000). A Combinatorial Auction for Collaborative Planning. *Proceedings of International Conference of Multi Agnents Systems.*

[25] Jones, G. E., Browning, B., Dias, B. M., Argall, B., & Veloso, M. M. (2006). Dynamically Formed Heterogenous Robot Teams Refroming Tightly Coordinated Tasks. *Robotics and Automation* , 570-575.

[26] Kalra, N., Ferguson, D., & Stentz, A. (2007). A Generalized Framework for Solving Tightly-coupled Multirobot Planing Problems. *IEEE International Conference on Robotics and Automation*, (pp. 3359-3364). Roma.

[27] Kaminka, G., & Tambe, M. (2000). Robust Agent Teams via Socially-Attentive Monitoring. *Journal of Artificial Intelligence Reseach , vol.12*, 105-147.

[28] Koening, S., Tovey, C., Zheng, X., & Sungur, I. (2007). Sequential bundle-bid single-sale aution algorithms for decentralized control. *Proceedings of the International Conference on Artifical Intelligence*, (pp. 1359-1365).

[29] Koes, M., Nourbakhsh, I., & Sycara, K. (2005). Heterogeneous Multirobot Coordination with Spatial and Temporal Constraints. *AAAI* .

[30] Lerman, K., Jones, C., Galstyan, A., & Matarić, M. J. (2011). Analysis of Dynamic Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research* , 590-600.

[31] Lin, L., & Zheng, Z. (2005). Combinatorial Bids based Multi-robot Task Allocation Method. *Proceedings of the IEEE International Conference on Robotics and Automation.* Barcelona.

[32] McAfee, P. R. (1987). Auctions and Bidding. *Journal of Ecomonic Literature* , 699-738.

[33] Miyata, N., Ota, J., Aria, T., & Asama, H. (2002). Cooperative Transport by Multiple Mobile Robots in Unknown Static Environments Associated With Real-Time Task Assignment. *IEEE Transaction on Robotics and Automation , vol.18*, 769-780.

[34] Mosteo, A., Montano, L., & Lagoudakis, M. (2008). Multi-Robot Routing under Limited Communication Range. *IEEE International Conference on Robotics and Automation*, (pp. 1531-1536). Pasadena.

[35] Myriam, A., Chao, W., & Mittu, R. (2005). Design and evaluation of distributed role algorithms in open environments. *Reaseach Lab Washinhton Naval* .

[36] Nanjanath, M. (2010). *Repeated Auction for Robust Task Excecution by a Rbot Team.* Minnesota: University of Minnesota.

[37] Nathan, M., Zavlanos, M. M., Kumar, V., & Pappas, G. J. (2008). Distributed Multi-Robot Task Assignment and Formation Control. *IEEE Transaction on Robotics and Automaation* , 255-260.

[38] Parker, L. (1998). ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation. *IEEE Transactions on Robotics and Automation* , 220-240.

[39] Portucal, D., & Siassipour, B. (2010). Study of Auction-Based Methods for Multi-Robot Coordination.

[40] Roy, N., & Dudek, G. (2001). Collaborative Robot Exploration and Rendezvous: Algorithms, Performance Bounds and Observations. *Autonomus Robots* , 117-136.

[41] Sandholm, T., & Lesser, V. (1996). Advantages Of A Leveled Commitment Contracting Protocol. *Proceeding of the 30th National Conference* , (pp. 126-133).

[42] Sandholm, T., & Suri, S. (1999). Algorithms for Optimizing Leveled Commitment Contracts. *International Joint Conference on Artificial Intelligence*, (pp. 535-540).

[43] Sariel-Talay, S., Blach, T. R., & Erdogan, N. (2011). A Generic Framework for Distibuted Multirobot Cooperation. *Intellignce Robot Systems* , 323-358.

[44] Simonin, O., & Grunder, O. (2009). A cooperative multi-robot architecture for moving a paralyzed robot. *Mechatronics* , 462-470.

[45] Smith, R. (1980). The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers* , 1104-1113.

[46] Stentz, A., Dias, B. M., Zlot, R., & Kalra, N. (2004). Market-based Approaches for Coordination of Multi-robot Teams at Different Granularities of Interaciton. *Robotics Institute* .

[47] Stone, P., & Veloso, M. (1999). Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence* , 241-273.

[48] Tang, F., & Parker, L. E. (2005). Distributed Multi-Robot Coalitions through ASyMTEe-D. *IEEE International Conference on Intelligent Robots and Systems.* Edmonton.

[49] Zlot, R. (2006). *An Auction-Based Approach to Complex Task Allocation for Multirobot Teams.* Pittsburgh, Pennsylvania: The Robotics Institue Carnegie Mellon University.

[50] Zlot, R. (2005). Complex Task Allocation For Multiple Robots. *Robotics and Automation* , 1515-1522.

[51] Zlot, R., & Stentz, A. (2006, January). Market-based Multirobot Coordination for Complex Tasks. *International Journal of Robotics Research* , 230-251.

[52] Zulueta, G., Torres, S., & Tur, M. (2010). Guiding and regrouping people missions in urban areas using cooperative multi-task allocation.